СПРАВОЧНИК по языку программирования MQL5

для клиентского терминала MetaTrader 5

ИЗУЧИТЕ язык MQL5
и РЕШАЙТЕ любые задачи:

- Создание собственных индикаторов технического анализа любой сложности
- Автотрейдинг - автоматизация торговой системы для работы на разнообразных финансовых рынках
- Разработка аналитических инструментов на основе математических достижений и классических методов
- Написание информационно-торговых систем для широкого круга задач (трейдинг, мониторинг, сигналы и т.д.)

© 2000-2025, MetaQuotes Ltd

Содержание

Справочник MQL5 72

1 Основы языка ... 75

Синтаксис ... 76
Комментарии ... 77
Идентификаторы ... 78
Зарезервированные слова ... 79
Типы данных ... 81
Целые типы ... 82
Типы char, short, int и long ... 84
Символьные константы ... 88
Тип datetime ... 92
Тип color ... 93
Тип bool ... 94
Перечисления ... 95
Вещественные типы (double, float) ... 97
Комплексное число (complex) ... 103
Тип string ... 104
Структуры, классы и интерфейсы ... 107
Объект динамического массива ... 135
Матрицы и векторы ... 137
Приведение типов ... 145
Тип void и константа NULL ... 150
Пользовательские типы ... 151
Указатели объектов ... 161
Ссылки. Модификатор & и ключевое слово this ... 165
Операции и выражения ... 167
Выражения ... 168
Арифметические операции ... 169
Операции присваивания ... 170
Операции отношения ... 171
Логические операции ... 172
Побитовые операции ... 174
Другие операции ... 177
Приоритеты и порядок операций ... 181
Операторы ... 183
Составной оператор ... 185
Оператор-выражение ... 186
Оператор возврата return ... 187
Условный оператор if-else ... 188
Условный оператор ?: ... 189
Оператор-переключатель switch ... 191
Оператор цикла while ... 193
Оператор цикла for ... 194
Оператор цикла do while ... 196
Оператор завершения break ... 197
Оператор продолжения continue ... 198
Оператор произведения матриц ... 199
Оператор создания объекта new ... 201
Оператор уничтожения объекта delete ... 203
Функции ... 204
Вызов функции ... 206
Передача параметров ... 207
Перегрузка функций ... 210

© 2000-2025, MetaQuotes Ltd.

Перегрузка операций ... 213
Описание внешних функций ... 227
Экспортирование функций ... 229
Функции обработки событий ... 230
Переменные ... 243
Локальные переменные ... 247
Формальные параметры ... 249
Статические переменные ... 251
Глобальные переменные ... 253
Input переменные ... 254
Extern переменные ... 261
Инициализация переменных ... 262
Область видимости и время жизни переменных ... 264
Создание и уничтожение объектов ... 266
Препроцессор ... 269
Макроподстановка (#define) ... 271
Свойства программ (#property) ... 274
Включение файлов (#include) ... 280
Импорт функций (#import) ... 281
Условная компиляция (#ifdef, #ifndef, #else, #endif) ... 284
Объектно-ориентированное программирование ... 286
Инкапсуляция и расширяемость типов ... 288
Наследование ... 291
Полиморфизм ... 296
Перегрузка ... 300
Виртуальные функции ... 301
Статические члены класса ... 305
Шаблоны функций ... 309
Шаблоны классов ... 314
Абстрактные классы ... 319
Пространства имен ... 321
2 Константы, перечисления и структуры ... 325
Константы графиков ... 326
Типы событий графика ... 327
Периоды графиков ... 335
Свойства графиков ... 337
Позиционирование графика ... 352
Отображение графиков ... 353
Примеры работы с графиком ... 355
Константы объектов ... 414
Типы объектов ... 415
OBJ_VLINE ... 417
OBJ_HLINE ... 422
OBJ_TREND ... 427
OBJ_TRENDBYANGLE ... 434
OBJ_CYCLES ... 440
OBJ_ARROWED_LINE ... 446
OBJ_CHANNEL ... 452
OBJ_STDDEVCHANNEL ... 459
OBJ_REGRESSION ... 466
OBJ_PITCHFORK ... 472
OBJ_GANNLINE ... 480
OBJ_GANNFAN ... 487
OBJ_GANNGRID ... 494
OBJ_FIBO ... 501
OBJ_FIBOTIMES ... 508
OBJ_FIBOFAN ... 515
OBJ_FIBOARC ... 522

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOCHANNEL ... 529
OBJ_EXPANSION ... 537
OBJ_ELLIOTWAVE5 ... 545
OBJ_ELLIOTWAVE3 ... 553
OBJ_RECTANGLE ... 560
OBJ_TRIANGLE ... 566
OBJ_ELLIPSE ... 573
OBJ_ARROW_THUMB_UP ... 579
OBJ_ARROW_THUMB_DOWN ... 585
OBJ_ARROW_UP ... 591
OBJ_ARROW_DOWN ... 597
OBJ_ARROW_STOP ... 603
OBJ_ARROW_CHECK ... 609
OBJ_ARROW_LEFT_PRICE ... 615
OBJ_ARROW_RIGHT_PRICE ... 620
OBJ_ARROW_BUY ... 625
OBJ_ARROW_SELL ... 630
OBJ_ARROW ... 635
OBJ_TEXT ... 641
OBJ_LABEL ... 647
OBJ_BUTTON ... 655
OBJ_CHART ... 662
OBJ_BITMAP ... 669
OBJ_BITMAP_LABEL ... 676
OBJ_EDIT ... 683
OBJ_EVENT ... 690
OBJ_RECTANGLE_LABEL ... 695
Свойства объектов ... 701
Способы привязки объектов ... 734
Угол привязки ... 739
Видимость объектов ... 742
Уровни волн Эллиотта ... 745
Объекты Ганна ... 746
Набор Web-цветов ... 748
Wingdings ... 750
Константы индикаторов ... 751
Ценовые константы ... 752
Методы скользящих ... 755
Линии индикаторов ... 756
Стили рисования ... 758
Свойства пользовательских индикаторов ... 764
Типы индикаторов ... 771
Идентификаторы типов данных ... 773
Состояние окружения ... 774
Состояние клиентского терминала ... 775
Информация о запущенной MQL5-программе ... 786
Информация об инструменте ... 791
Информация о счете ... 906
Статистика тестирования ... 915
Торговые константы ... 921
Информация об исторических данных по инструменту ... 922
Свойства ордеров ... 923
Свойства позиций ... 943
Свойства сделок ... 948
Типы торговых операций ... 953
Типы торговых транзакций ... 963
Виды заявок в стакане цен ... 966
Свойства сигналов ... 967

© 2000-2025, MetaQuotes Ltd.

Именованные константы **969**
Предопределенные макроподстановки 970
Математические константы 976
Константы числовых типов 978
Причины деинициализации 982
Проверка указателя объекта 984
Прочие константы 985
Структуры данных **989**
Структура даты 990
Структура входных параметров индикатора 991
Структура исторических данных 992
Структура стакана цен 993
Структура торгового запроса 994
Структура результата проверки торгового запроса 1007
Структура результата торгового запроса 1008
Структура торговой транзакции 1012
Структура для получения текущих цен 1020
Структуры экономического календаря 1022
Коды ошибок и предупреждений **1031**
Коды возврата торгового сервера 1032
Предупреждения компилятора 1036
Ошибки компиляции 1039
Ошибки времени выполнения 1050
Константы ввода/вывода **1065**
Флаги открытия файлов 1066
Свойства файлов 1068
Позиционирование внутри файла 1069
Использование кодовой страницы 1070
MessageBox 1071
3 Программы MQL5 ..... **1073**
Выполнение программ **1074**
Разрешение на торговлю **1082**
События клиентского терминала **1086**
Ресурсы **1089**
Вызов импортируемых функций **1101**
Ошибки выполнения **1103**
Тестирование торговых стратегий **1104**
4 Предопределенные переменные **1131**
_AppliedTo **1132**
_Digits **1134**
_Point **1135**
_LastError **1136**
_Period **1137**
_RandomSeed **1138**
_StopFlag **1139**
_Symbol **1140**
_UninitReason **1141**
_IsX64 ..... **1142**
5 Общие функции **1143**
Alert **1145**
CheckPointer **1150**
Comment **1152**
CryptEncode **1154**
CryptDecode **1156**
DebugBreak **1158**
ExpertRemove **1159**
GetPointer **1161**

© 2000-2025, MetaQuotes Ltd.

GetTickCount ... 1165
GetTickCount64 ... 1167
GetMicrosecondCount ... 1169
MessageBox ... 1171
PeriodSeconds ... 1176
PlaySound ... 1178
Print ... 1181
PrintFormat ... 1183
ResetLastError ... 1189
ResourceCreate ... 1190
ResourceFree ... 1194
ResourceReadImage ... 1197
ResourceSave ... 1201
SetReturnError ... 1206
SetUserError ... 1208
Sleep ... 1209
TerminalClose ... 1211
TesterHideIndicators ... 1213
TesterStatistics ... 1215
TesterStop ... 1217
TesterDeposit ... 1219
TesterWithdrawal ... 1222
TranslateKey ... 1225
ZeroMemory ... 1226

6 Операции с массивами ... 1229

ArrayBsearch ... 1231
ArrayCopy ... 1235
ArrayCompare ... 1240
ArrayFree ... 1243
ArrayGetAsSeries ... 1252
ArrayInitialize ... 1255
ArrayFill ... 1257
ArrayIsDynamic ... 1259
ArrayIsSeries ... 1261
ArrayMaximum ... 1263
ArrayMinimum ... 1274
ArrayPrint ... 1285
ArrayRange ... 1288
ArrayResize ... 1289
ArrayInsert ... 1292
ArrayRemove ... 1295
ArrayReverse ... 1297
ArraySetAsSeries ... 1299
ArraySize ... 1302
ArraySort ... 1304
ArraySwap ... 1309
ArrayToFP16 ... 1311
ArrayToFP8 ... 1314
ArrayFromFP16 ... 1317
ArrayFromFP8 ... 1320

7 Методы матриц и векторов ... 1323

Типы матриц и векторов ... 1331
Перечисления ... 1333
Инициализация ... 1338
Assign ... 1341
CopyIndicatorBuffer ... 1343
CopyRates ... 1345
CopyTicks ... 1350

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange ... 1353
Eye ... 1355
Identity ... 1357
Ones ... 1359
Zeros ... 1360
Full ... 1361
Tri ... 1362
Init ... 1363
Fill ... 1365
Random ... 1366
Манипуляции ... 1368
HasNan ... 1370
Transpose ... 1371
TransposeConjugate ... 1373
TriL ... 1374
TriU ... 1375
Diag ... 1376
Row ... 1378
Col ... 1380
Copy ... 1382
Concat ... 1384
Compare ... 1386
CompareByDigits ... 1388
CompareEqual ... 1390
Flat ... 1393
Clip ... 1395
Reshape ... 1396
Resize ... 1398
Set ... 1400
SwapRows ... 1402
SwapCols ... 1403
Split ... 1404
Hsplit ... 1406
Vsplit ... 1408
ArgSort ... 1410
Sort ... 1411
Операции ... 1413
Математические операции ... 1415
Математические функции ... 1416
Произведения ... 1417
MatMul ... 1418
GeMM ... 1423
Power ... 1427
Dot ... 1430
Kron ... 1432
Inner ... 1434
Outer ... 1436
CorrCoef ... 1439
Cov ... 1443
Correlate ... 1446
Convolve ... 1449
Преобразования ... 1452
Cholesky ... 1453
Eig ... 1454
EigVals ... 1457
LU ... 1458
LUP ... 1460
QR ... 1462

© 2000-2025, MetaQuotes Ltd.

SVD ... 1464
Статистика ... 1467
ArgMax ... 1468
ArgMin ... 1469
Max ... 1470
Min ... 1471
Ptp ... 1472
Sum ... 1473
Prod ... 1474
CumSum ... 1476
CumProd ... 1478
Percentile ... 1480
Quantile ... 1482
Median ... 1484
Mean ... 1486
Average ... 1487
Std ... 1489
Var ... 1491
LinearRegression ... 1493
Характеристики ... 1496
Rows ... 1497
Cols ... 1498
Size ... 1499
Norm ... 1500
Cond ... 1503
Det ... 1506
SLogDet ... 1508
Rank ... 1509
Trace ... 1512
Spectrum ... 1513
Решения ... 1514
Solve ... 1515
LstSq ... 1516
Inv ... 1517
PInv ... 1519
Машинное обучение ... 1520
Activation ... 1526
Derivative ... 1530
Loss ... 1532
LossGradient ... 1534
RegressionMetric ... 1536
ConfusionMatrix ... 1538
ConfusionMatrixMultilabel ... 1540
ClassificationMetric ... 1542
ClassificationScore ... 1545
PrecisionRecall ... 1549
ReceiverOperatingCharacteristic ... 1553
OpenBLAS ... 1557
Singular Value Decomposition ... 1560
SingularValueDecompositionDC ... 1561
SingularValueDecompositionQR ... 1563
SingularValueDecompositionQRPivot ... 1565
SingularValueDecompositionBisect ... 1569
SingularValueDecompositionJacobiHigh ... 1572
SingularValueDecompositionJacobiLow ... 1577
SingularValueDecompositionBidiagDC ... 1580
SingularValueDecompositionBidiagBisect ... 1583
SingularValueDecompositionBidiagQR ... 1586

© 2000-2025, MetaQuotes Ltd.

Eigen Values ... 1588
General Matrices ... 1593
EigenSolver ... 1596
EigenSolverX ... 1598
EigenSolverShur ... 1602
EigenSolver2 ... 1604
EigenSolver2X ... 1607
EigenSolver2Shur ... 1612
EigenSolver2Blocked ... 1615
EigenSolver2ShurBlocked ... 1618
Symmetric Matrices ... 1621
EigenSymmetricDC ... 1623
EigenSymmetricQR ... 1625
EigenSymmetricRobust ... 1627
EigenSymmetricBisect ... 1630
EigenSymmetricDC2s ... 1633
EigenSymmetricQR2s ... 1635
EigenSymmetricRobust2s ... 1637
EigenSymmetricBisect2s ... 1640
EigenSymmetric2DC ... 1643
EigenSymmetric2QR ... 1646
EigenSymmetric2Bisect ... 1649
Tridiagonal Matrices ... 1653
EigenTridiagonalDC ... 1655
EigenTridiagonalQR ... 1657
EigenTridiagonalRobust ... 1659
EigenTridiagonalBisect ... 1662
EigenTridiagonalQL ... 1665
EigenTridiagonalDCQ ... 1666
EigenTridiagonalQRQ ... 1668
EigenTridiagonalPosDefQ ... 1670
Orthogonal Factorizations ... 1672
FactorizationQR ... 1673
FactorizationQRNonNeg ... 1675
FactorizationQRPivot ... 1677
FactorizationLQ ... 1679
FactorizationQL ... 1681
FactorizationRQ ... 1683
Linear Equations ... 1685
LinearEquationsSolution ... 1687
LinearEquationsSolutionTriangular ... 1689
LinearEquationsSolutionSy ... 1691
LinearEquationsSolutionComplexSy ... 1693
LinearEquationsSolutionSyPD ... 1694
LinearEquationsSolutionGeTrid ... 1696
LinearEquationsSolutionSyTridPD ... 1698
Factorizations ... 1700
FactorizationPLU ... 1702
FactorizationPLUGeTrid ... 1704
FactorizationLDL ... 1706
FactorizationLDLSyTridPD ... 1708
FactorizationCholesky ... 1710
FactorizationCholeskySyPS ... 1711
Dynamic Mode Decomposition ... 1713
DynamicModeDecomposition ... 1714
DynamicModeDecompositionQR ... 1720
Singular Spectrum Analysis ... 1727
SingularSpectrumAnalysisSpectrum ... 1728

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisForecast ... 1729
SingularSpectrumAnalysisReconstructComponents ... 1730
SingularSpectrumAnalysisReconstructSeries ... 1731
Reductions ... 1732
ReduceToBidiagonal ... 1733
ReflectBidiagonalToQP ... 1736
ReduceSymmetricToTridiagonal ... 1738
ReflectTridiagonalToQ ... 1740
8 Преобразование данных ... 1742
CharToString ... 1744
CharArrayToString ... 1746
CharArrayToStruct ... 1748
StructToCharArray ... 1752
ColorToARGB ... 1755
ColorToString ... 1757
DoubleToString ... 1758
EnumToString ... 1759
IntegerToString ... 1761
ShortToString ... 1763
ShortArrayToString ... 1765
TimeToString ... 1767
NormalizeDouble ... 1770
StringToCharArray ... 1772
StringToColor ... 1774
StringToDouble ... 1775
StringToInteger ... 1776
StringToShortArray ... 1777
StringToTime ... 1779
StringFormat ... 1782
9 Математические функции ... 1786
MathAbs ... 1788
MathArccos ... 1789
MathArcsin ... 1792
MathArctan ... 1795
MathArctan2 ... 1798
MathClassify ... 1802
MathCeil ... 1804
MathCos ... 1806
MathExp ... 1809
MathFloor ... 1812
MathLog ... 1814
MathLog10 ... 1818
MathMax ... 1822
MathMin ... 1824
MathMod ... 1826
MathPow ... 1827
MathRand ... 1831
MathRound ... 1832
MathSin ... 1834
MathSqrt ... 1837
MathSrand ... 1841
MathTan ... 1844
MathIsValidNumber ... 1847
MathExpm1 ... 1848
MathLog1p ... 1851
MathArccosh ... 1855
MathArcsinh ... 1858
MathArctanh ... 1861

© 2000-2025, MetaQuotes Ltd.

MathCosh ... 1864
MathSinh ... 1867
MathTanh ... 1870
MathSwap ... 1873
10 Строковые функции ... 1876
StringAdd ... 1877
StringBufferLen ... 1879
StringCompare ... 1880
StringConcatenate ... 1882
StringFill ... 1884
StringFind ... 1885
StringGetCharacter ... 1888
StringInit ... 1890
StringLen ... 1891
StringSetLength ... 1892
StringReplace ... 1893
StringReserve ... 1894
StringSetCharacter ... 1896
StringSplit ... 1898
StringSubstr ... 1900
StringToLower ... 1901
StringToUpper ... 1902
StringTrimLeft ... 1903
StringTrimRight ... 1904
11 Дата и время ... 1905
TimeCurrent ... 1906
TimeTradeServer ... 1908
TimeLocal ... 1910
TimeGMT ... 1912
TimeDaylightSavings ... 1914
TimeGMTOffset ... 1915
TimeToStruct ... 1916
StructToTime ... 1917
12 Информация о счете ... 1919
AccountInfoDouble ... 1920
AccountInfoInteger ... 1921
AccountInfoString ... 1923
13 Проверка состояния ... 1924
GetLastError ... 1925
IsStopped ... 1926
UninitializeReason ... 1927
TerminalInfoInteger ... 1929
TerminalInfoDouble ... 1930
TerminalInfoString ... 1931
MQLInfoInteger ... 1932
MQLInfoString ... 1933
Symbol ... 1934
Period ... 1935
Digits ... 1936
Point ... 1937
14 Обработка событий ... 1938
OnStart ... 1940
OnInit ... 1943
OnDeinit ... 1946
OnTick ... 1949
OnCalculate ... 1955

© 2000-2025, MetaQuotes Ltd.

OnTimer ... 1959
OnTrade ... 1962
OnTradeTransaction ... 1967
OnBookEvent ... 1973
OnChartEvent ... 1977
OnTester ... 1984
OnTesterInit ... 1991
OnTesterDeinit ... 1998
OnTesterPass ... 1999
15 Получение рыночной информации ... 2000
SymbolsTotal ... 2001
SymbolExist ... 2002
SymbolName ... 2004
SymbolSelect ... 2006
SymbolIsSynchronized ... 2008
SymbolInfoDouble ... 2009
SymbolInfoInteger ... 2011
SymbolInfoString ... 2013
SymbolInfoMarginRate ... 2015
SymbolInfoTick ... 2017
SymbolInfoSessionQuote ... 2018
SymbolInfoSessionTrade ... 2020
MarketBookAdd ... 2022
MarketBookRelease ... 2024
MarketBookGet ... 2026
16 Экономический календарь ... 2027
CalendarCountryById ... 2028
CalendarEventById ... 2030
CalendarValueById ... 2033
CalendarCountries ... 2036
CalendarEventByCountry ... 2038
CalendarEventByCurrency ... 2040
CalendarValueHistoryByEvent ... 2042
CalendarValueHistory ... 2046
CalendarValueLastByEvent ... 2049
CalendarValueLast ... 2054
17 Доступ к таймсериям и индикаторам ... 2059
Направление индексации в массивах и таймсериях ... 2064
Организация доступа к данным ... 2068
SeriesInfoInteger ... 2078
Bars ... 2080
BarsCalculated ... 2083
IndicatorCreate ... 2085
IndicatorParameters ... 2088
IndicatorRelease ... 2090
CopyBuffer ... 2092
CopyRates ... 2097
CopySeries ... 2101
CopyTime ... 2105
CopyOpen ... 2108
CopyHigh ... 2111
CopyLow ... 2115
CopyClose ... 2118
CopyTickVolume ... 2121
CopyRealVolume ... 2125
CopySpread ... 2128
CopyTicks ... 2132

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange 2138
iBars 2140
iBarShift 2141
iClose 2144
iHigh 2146
iHighest 2148
iLow 2149
iLowest 2151
iOpen 2152
iTime 2154
iTickVolume 2157
iRealVolume 2159
iVolume 2161
iSpread 2163

18 Пользовательские символы 2165

CustomSymbolCreate 2167
CustomSymbolDelete 2170
CustomSymbolSetInteger 2172
CustomSymbolSetDouble 2176
CustomSymbolSetString 2180
CustomSymbolSetMarginRate 2184
CustomSymbolSetSessionQuote 2189
CustomSymbolSetSessionTrade 2194
CustomRatesDelete 2199
CustomRatesReplace 2204
CustomRatesUpdate 2209
CustomTicksAdd 2214
CustomTicksDelete 2222
CustomTicksReplace 2230
CustomBookAdd 2239

19 Операции с графиками 2242

ChartApplyTemplate 2244
ChartSaveTemplate 2247
ChartWindowFind 2252
ChartTimePriceToXY 2254
ChartXYToTimePrice 2256
ChartOpen 2258
ChartFirst 2260
ChartNext 2261
ChartClose 2262
ChartSymbol 2264
ChartPeriod 2265
ChartRedraw 2266
ChartSetDouble 2269
ChartSetInteger 2271
ChartSetString 2273
ChartGetDouble 2275
ChartGetInteger 2277
ChartGetString 2279
ChartNavigate 2281
ChartID 2284
ChartIndicatorAdd 2286
ChartIndicatorDelete 2290
ChartIndicatorGet 2293
ChartIndicatorName 2295
ChartIndicatorsTotal 2297
ChartWindowOnDropped 2299
ChartPriceOnDropped 2300

© 2000-2025, MetaQuotes Ltd.

ChartTimeOnDropped 2301
ChartXOnDropped 2302
ChartYOnDropped 2303
ChartSetSymbolPeriod 2304
ChartScreenShot 2306
20 Торговые функции 2309
OrderCalcMargin 2311
OrderCalcProfit 2315
OrderCheck 2318
OrderSend 2322
OrderSendAsync 2327
PositionsTotal 2338
PositionGetSymbol 2339
PositionSelect 2341
PositionSelectByTicket 2343
PositionGetDouble 2345
PositionGetInteger 2349
PositionGetString 2351
PositionGetTicket 2354
OrdersTotal 2356
OrderGetTicket 2357
OrderSelect 2359
OrderGetDouble 2363
OrderGetInteger 2367
OrderGetString 2373
HistorySelect 2376
HistorySelectByPosition 2378
HistoryOrderSelect 2382
HistoryOrdersTotal 2385
HistoryOrderGetTicket 2386
HistoryOrderGetDouble 2388
HistoryOrderGetInteger 2392
HistoryOrderGetString 2395
HistoryDealSelect 2398
HistoryDealsTotal 2401
HistoryDealGetTicket 2402
HistoryDealGetDouble 2404
HistoryDealGetInteger 2409
HistoryDealGetString 2412
21 Управление сигналами 2416
SignalBaseGetDouble 2417
SignalBaseGetInteger 2418
SignalBaseGetString 2419
SignalBaseSelect 2420
SignalBaseTotal 2421
SignalInfoGetDouble 2422
SignalInfoGetInteger 2423
SignalInfoGetString 2424
SignalInfoSetDouble 2425
SignalInfoSetInteger 2426
SignalSubscribe 2427
SignalUnsubscribe 2428
22 Сетевые функции 2429
SocketCreate 2431
SocketClose 2434
SocketConnect 2437
SocketIsConnected 2441

© 2000-2025, MetaQuotes Ltd.

SocketIsReadable ... 2445
SocketIsWritable ... 2448
SocketTimeouts ... 2452
SocketRead ... 2454
SocketSend ... 2458
SocketTlsHandshake ... 2462
SocketTlsCertificate ... 2467
SocketTlsRead ... 2471
SocketTlsReadAvailable ... 2475
SocketTlsSend ... 2480
WebRequest ... 2484
SendFTP ... 2487
SendMail ... 2489
SendNotification ... 2491
23 Глобальные переменные терминала ... 2493
GlobalVariableCheck ... 2494
GlobalVariableTime ... 2495
GlobalVariableDel ... 2496
GlobalVariableGet ... 2497
GlobalVariableName ... 2498
GlobalVariableSet ... 2499
GlobalVariablesFlush ... 2500
GlobalVariableTemp ... 2501
GlobalVariableSetOnCondition ... 2502
GlobalVariablesDeleteAll ... 2503
GlobalVariablesTotal ... 2504
24 Файловые операции ... 2505
FileSelectDialog ... 2508
FileFindFirst ... 2511
FileFindNext ... 2513
FileFindClose ... 2515
FileIsExist ... 2517
FileOpen ... 2520
FileClose ... 2523
FileCopy ... 2524
FileDelete ... 2527
FileMove ... 2529
FileFlush ... 2531
FileGetInteger ... 2533
FileIsEnding ... 2536
FileIsLineEnding ... 2538
FileReadArray ... 2543
FileReadBool ... 2545
FileReadDatetime ... 2548
FileReadDouble ... 2551
FileReadFloat ... 2554
FileReadInteger ... 2557
FileReadLong ... 2561
FileReadNumber ... 2564
FileReadString ... 2569
FileReadStruct ... 2571
FileSeek ... 2575
FileSize ... 2578
FileTell ... 2580
FileWrite ... 2583
FileWriteArray ... 2586
FileWriteDouble ... 2589
FileWriteFloat ... 2592

© 2000-2025, MetaQuotes Ltd.

FileWriteInteger 2594
FileWriteLong 2597
FileWriteString 2599
FileWriteStruct 2602
FileLoad 2605
FileSave 2607
FolderCreate 2609
FolderDelete 2612
FolderClean 2615
25 Пользовательские индикаторы 2618
Стили индикаторов в примерах 2622
DRAW_NONE 2634
DRAW_LINE 2637
DRAW_SECTION 2641
DRAW_HISTOGRAM 2645
DRAW_HISTOGRAM2 2649
DRAW_ARROW 2653
DRAW_ZIGZAG 2658
DRAW_FILLING 2663
DRAW_BARS 2668
DRAW_CANDLES 2674
DRAW_COLOR_LINE 2681
DRAW_COLOR_SECTION 2686
DRAW_COLOR_HISTOGRAM 2692
DRAW_COLOR_HISTOGRAM2 2697
DRAW_COLOR_ARROW 2702
DRAW_COLOR_ZIGZAG 2708
DRAW_COLOR_BARS 2713
DRAW_COLOR_CANDLES 2720
Связь между свойствами индикатора и функциями 2727
SetIndexBuffer 2730
IndicatorSetDouble 2733
IndicatorSetInteger 2737
IndicatorSetString 2741
PlotIndexSetDouble 2744
PlotIndexSetInteger 2745
PlotIndexSetString 2749
PlotIndexGetInteger 2750
26 Графические объекты 2753
ObjectCreate 2755
ObjectName 2759
ObjectDelete 2760
ObjectsDeleteAll 2761
ObjectFind 2762
ObjectGetTimeByValue 2763
ObjectGetValueByTime 2764
ObjectMove 2765
ObjectsTotal 2766
ObjectSetDouble 2767
ObjectSetInteger 2771
ObjectSetString 2774
ObjectGetDouble 2776
ObjectGetInteger 2778
ObjectGetString 2780
TextSetFont 2782
TextOut 2785
TextGetSize 2789

© 2000-2025, MetaQuotes Ltd.

27 Технические индикаторы 2790
iAC 2793
iAD 2798
iADX 2803
iADXWilder 2808
iAlligator 2813
iAMA 2820
iAO 2825
iATR 2830
iBearsPower 2835
iBands 2840
iBullsPower 2846
iCCI 2851
iChaikin 2856
iCustom 2861
iDEMA 2865
iDeMarker 2870
iEnvelopes 2875
iForce 2881
iFractals 2886
iFrAMA 2891
iGator 2896
iIchimoku 2903
iBWMFI 2910
iMomentum 2915
iMFI 2920
iMA 2925
iOsMA 2930
iMACD 2935
iOBV 2941
iSAR 2946
iRSI 2951
iRVI 2956
iStdDev 2961
iStochastic 2966
iTEMA 2972
iTriX 2977
iWPR 2982
iVIDyA 2987
iVolumes 2992
28 Работа с результатами оптимизации 2997
FrameFirst 2999
FrameFilter 3000
FrameNext 3004
FrameInputs 3005
FrameAdd 3006
ParameterGetRange 3007
ParameterSetRange 3010
29 Работа с событиями 3012
EventSetMillisecondTimer 3013
EventSetTimer 3014
EventKillTimer 3015
EventChartCustom 3016
30 Работа с OpenCL 3022
CLHandleType 3025
CLGetInfoInteger 3026
CLGetInfoString 3029

© 2000-2025, MetaQuotes Ltd.

CLContextCreate ... 3032
CLContextFree ... 3033
CLGetDeviceInfo ... 3034
CLProgramCreate ... 3039
CLProgramFree ... 3044
CLKernelCreate ... 3045
CLKernelFree ... 3046
CLSetKernelArg ... 3047
CLSetKernelArgMem ... 3048
CLSetKernelArgMemLocal ... 3049
CLBufferCreate ... 3050
CLBufferFree ... 3051
CLBufferWrite ... 3052
CLBufferRead ... 3057
CLExecute ... 3061
CLExecutionStatus ... 3063
31 Работа с базами данных ... 3064
DatabaseOpen ... 3068
DatabaseClose ... 3070
DatabaseImport ... 3071
DatabaseExport ... 3074
DatabasePrint ... 3080
DatabaseTableExists ... 3085
DatabaseExecute ... 3086
DatabasePrepare ... 3098
DatabaseReset ... 3107
DatabaseBind ... 3113
DatabaseBindArray ... 3118
DatabaseRead ... 3123
DatabaseReadBind ... 3124
DatabaseFinalize ... 3128
DatabaseTransactionBegin ... 3129
DatabaseTransactionCommit ... 3134
DatabaseTransactionRollback ... 3135
DatabaseColumnsCount ... 3136
DatabaseColumnName ... 3137
DatabaseColumnType ... 3138
DatabaseColumnSize ... 3139
DatabaseColumnText ... 3140
DatabaseColumnInteger ... 3141
DatabaseColumnLong ... 3142
DatabaseColumnDouble ... 3143
DatabaseColumnBlob ... 3144
32 Работа с DirectX ... 3145
DXContextCreate ... 3147
DXContextSetSize ... 3148
DXContextGetSize ... 3149
DXContextClearColors ... 3150
DXContextClearDepth ... 3151
DXContextGetColors ... 3152
DXContextGetDepth ... 3153
DXBufferCreate ... 3154
DXTextureCreate ... 3155
DXInputCreate ... 3161
DXInputSet ... 3162
DXShaderCreate ... 3163
DXShaderSetLayout ... 3164
DXShaderInputsSet ... 3165

© 2000-2025, MetaQuotes Ltd.

DXShaderTexturesSet ... 3166
DXDraw ... 3167
DXDrawIndexed ... 3168
DXPrimiveTopologySet ... 3169
DXBufferSet ... 3170
DXShaderSet ... 3171
DXHandleType ... 3172
DXRelease ... 3173
33 Python интеграция ... 3174
initialize ... 3180
login ... 3182
shutdown ... 3185
version ... 3186
last_error ... 3188
account_info ... 3190
terminal_info ... 3193
symbols_total ... 3196
symbols_get ... 3197
symbol_info ... 3200
symbol_info_tick ... 3204
symbol_select ... 3206
market_book_add ... 3210
market_book_get ... 3211
market_book_release ... 3214
copy_rates_from ... 3215
copy_rates_from_pos ... 3219
copy_rates_range ... 3222
copy_ticks_from ... 3225
copy_ticks_range ... 3229
orders_total ... 3232
orders_get ... 3233
order_calc_margin ... 3236
order_calc_profit ... 3239
order_check ... 3242
order_send ... 3246
positions_total ... 3251
positions_get ... 3252
history_orders_total ... 3255
history_orders_get ... 3257
history_deals_total ... 3260
history_deals_get ... 3262
34 ONNX модели ... 3266
Поддержка ONNX ... 3267
Преобразование форматов ... 3269
Автоконвертация данных ... 3270
Создание модели ... 3274
Запуск модели ... 3282
Проверка в тестере ... 3288
OnnxCreate ... 3293
OnnxCreateFromBuffer ... 3294
OnnxRelease ... 3295
OnnxRun ... 3296
OnnxGetInputCount ... 3299
OnnxGetOutputCount ... 3300
OnnxGetInputName ... 3301
OnnxGetOutputName ... 3302
OnnxGetInputTypeInfo ... 3303
OnnxGetOutputTypeInfo ... 3304

© 2000-2025, MetaQuotes Ltd.

OnnxSetInputShape ... 3305
OnnxSetOutputShape ... 3306
Структуры данных ... 3307
35 Стандартная библиотека ... 3312
Математика ... 3313
Статистика ... 3314
Статистические характеристики ... 3317
MathMean ... 3318
MathVariance ... 3319
MathSkewness ... 3320
MathKurtosis ... 3321
MathMoments ... 3322
MathMedian ... 3323
MathStandardDeviation ... 3324
MathAverageDeviation ... 3325
Нормальное распределение ... 3326
MathProbabilityDensityNormal ... 3330
MathCumulativeDistributionNormal ... 3332
MathQuantileNormal ... 3334
MathRandomNormal ... 3336
MathMomentsNormal ... 3337
Логнормальное распределение ... 3338
MathProbabilityDensityLognormal ... 3342
MathCumulativeDistributionLognormal ... 3344
MathQuantileLognormal ... 3346
MathRandomLognormal ... 3348
MathMomentsLognormal ... 3349
Бета-распределение ... 3350
MathProbabilityDensityBeta ... 3354
MathCumulativeDistributionBeta ... 3356
MathQuantileBeta ... 3358
MathRandomBeta ... 3360
MathMomentsBeta ... 3361
Нецентральное бета-распределение ... 3362
MathProbabilityDensityNoncentralBeta ... 3366
MathCumulativeDistributionNoncentralBeta ... 3368
MathQuantileNoncentralBeta ... 3370
MathRandomNoncentralBeta ... 3372
MathMomentsNoncentralBeta ... 3373
Гамма-распределение ... 3374
MathProbabilityDensityGamma ... 3378
MathCumulativeDistributionGamma ... 3380
MathQuantileGamma ... 3382
MathRandomGamma ... 3384
MathMomentsGamma ... 3385
Распределение хи-квадрат ... 3386
MathProbabilityDensityChiSquare ... 3390
MathCumulativeDistributionChiSquare ... 3392
MathQuantileChiSquare ... 3394
MathRandomChiSquare ... 3396
MathMomentsChiSquare ... 3397
Нецентральное распределение хи-квадрат ... 3398
MathProbabilityDensityNoncentralChiSquare ... 3402
MathCumulativeDistributionNoncentralChiSquare ... 3404
MathQuantileNoncentralChiSquare ... 3406
MathRandomNoncentralChiSquare ... 3408
MathMomentsNoncentralChiSquare ... 3409
Экспоненциальное распределение ... 3410

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityExponential ... 3414
MathCumulativeDistributionExponential ... 3416
MathQuantileExponential ... 3418
MathRandomExponential ... 3420
MathMomentsExponential ... 3421
F-распределение ... 3422
MathProbabilityDensityF ... 3426
MathCumulativeDistributionF ... 3428
MathQuantileF ... 3430
MathRandomF ... 3432
MathMomentsF ... 3433
Нецентральное F-распределение ... 3434
MathProbabilityDensityNoncentralF ... 3438
MathCumulativeDistributionNoncentralF ... 3440
MathQuantileNoncentralF ... 3442
MathRandomNoncentralF ... 3444
MathMomentsNoncentralF ... 3445
T-распределение ... 3446
MathProbabilityDensityT ... 3450
MathCumulativeDistributionT ... 3452
MathQuantileT ... 3454
MathRandomT ... 3456
MathMomentsT ... 3457
Нецентральное T-распределение ... 3458
MathProbabilityDensityNoncentralT ... 3462
MathCumulativeDistributionNoncentralT ... 3464
MathQuantileNoncentralT ... 3466
MathRandomNoncentralT ... 3468
MathMomentsNoncentralT ... 3469
Логистическое распределение ... 3470
MathProbabilityDensityLogistic ... 3474
MathCumulativeDistributionLogistic ... 3476
MathQuantileLogistic ... 3478
MathRandomLogistic ... 3480
MathMomentsLogistic ... 3481
Распределение Коши ... 3482
MathProbabilityDensityCauchy ... 3486
MathCumulativeDistributionCauchy ... 3488
MathQuantileCauchy ... 3490
MathRandomCauchy ... 3492
MathMomentsCauchy ... 3493
Равномерное распределение ... 3494
MathProbabilityDensityUniform ... 3498
MathCumulativeDistributionUniform ... 3500
MathQuantileUniform ... 3502
MathRandomUniform ... 3504
MathMomentsUniform ... 3505
Распределение Вейбулла ... 3506
MathProbabilityDensityWeibull ... 3510
MathCumulativeDistributionWeibull ... 3512
MathQuantileWeibull ... 3514
MathRandomWeibull ... 3516
MathMomentsWeibull ... 3517
Биномиальное распределение ... 3518
MathProbabilityDensityBinomial ... 3521
MathCumulativeDistributionBinomial ... 3523
MathQuantileBinomial ... 3525
MathRandomBinomial ... 3527

© 2000-2025, MetaQuotes Ltd.

MathMomentsBinomial ... 3528
Отрицательное биномиальное распределение ... 3529
MathProbabilityDensityNegativeBinomial ... 3532
MathCumulativeDistributionNegativeBinomial ... 3534
MathQuantileNegativeBinomial ... 3536
MathRandomNegativeBinomial ... 3538
MathMomentsNegativeBinomial ... 3539
Геометрическое распределение ... 3540
MathProbabilityDensityGeometric ... 3544
MathCumulativeDistributionGeometric ... 3546
MathQuantileGeometric ... 3548
MathRandomGeometric ... 3550
MathMomentsGeometric ... 3551
Гипергеометрическое распределение ... 3552
MathProbabilityDensityHypergeometric ... 3556
MathCumulativeDistributionHypergeometric ... 3558
MathQuantileHypergeometric ... 3560
MathRandomHypergeometric ... 3562
MathMomentsHypergeometric ... 3563
Распределение Пуассона ... 3564
MathProbabilityDensityPoisson ... 3568
MathCumulativeDistributionPoisson ... 3570
MathQuantilePoisson ... 3572
MathRandomPoisson ... 3574
MathMomentsPoisson ... 3575
Вспомогательные функции ... 3576
MathRandomNonZero ... 3581
MathMoments ... 3582
MathPowInt ... 3583
MathFactorial ... 3584
MathTrunc ... 3585
MathRound ... 3586
MathArctan2 ... 3588
MathGamma ... 3590
MathGammaLog ... 3591
MathBeta ... 3592
MathBetaLog ... 3593
MathBetaIncomplete ... 3594
MathGammaIncomplete ... 3595
MathBinomialCoefficient ... 3596
MathBinomialCoefficientLog ... 3597
MathHypergeometric2F2 ... 3598
MathSequence ... 3599
MathSequenceByCount ... 3600
MathReplicate ... 3601
MathReverse ... 3602
MathIdentical ... 3603
MathUnique ... 3604
MathQuickSortAscending ... 3605
MathQuickSortDescending ... 3606
MathQuickSort ... 3607
MathOrder ... 3608
MathBitwiseNot ... 3609
MathBitwiseAnd ... 3610
MathBitwiseOr ... 3611
MathBitwiseXor ... 3612
MathBitwiseShiftL ... 3613
MathBitwiseShiftR ... 3614

© 2000-2025, MetaQuotes Ltd.

MathCumulativeSum ... 3615
MathCumulativeProduct ... 3616
MathCumulativeMin ... 3617
MathCumulativeMax ... 3618
MathSin ... 3619
MathCos ... 3620
MathTan ... 3621
MathArcsin ... 3622
MathArccos ... 3623
MathArctan ... 3624
MathSinPi ... 3625
MathCosPi ... 3626
MathTanPi ... 3627
MathAbs ... 3628
MathCeil ... 3629
MathFloor ... 3630
MathSqrt ... 3631
MathExp ... 3632
MathPow ... 3633
MathLog ... 3634
MathLog2 ... 3635
MathLog10 ... 3636
MathLog1p ... 3637
MathDifference ... 3638
MathSample ... 3640
MathTukeySummary ... 3643
MathRange ... 3644
MathMin ... 3645
MathMax ... 3646
MathSum ... 3647
MathProduct ... 3648
MathStandardDeviation ... 3649
MathAverageDeviation ... 3650
MathMedian ... 3651
MathMean ... 3652
MathVariance ... 3653
MathSkewness ... 3654
MathKurtosis ... 3655
MathExpm1 ... 3656
MathSinh ... 3657
MathCosh ... 3658
MathTanh ... 3659
MathArcsinh ... 3660
MathArccosh ... 3661
MathArctanh ... 3662
MathSignif ... 3663
MathRank ... 3665
MathCorrelationPearson ... 3666
MathCorrelationSpearman ... 3667
MathCorrelationKendall ... 3668
MathQuantile ... 3669
MathProbabilityDensityEmpirical ... 3670
MathCumulativeDistributionEmpirical ... 3671
Нечеткая логика ... 3672
Функции принадлежности ... 3674
CConstantMembershipFunction ... 3676
GetValue ... 3678
CCompositeMembershipFunction ... 3679

© 2000-2025, MetaQuotes Ltd.

CompositionType ... 3681
MembershipFunctions ... 3681
GetValue ... 3681
CDifferencTwoSigmoidalMembershipFunction ... 3683
A1 ... 3685
A2 ... 3685
C1 ... 3686
C2 ... 3686
GetValue ... 3687
CGeneralizedBellShapedMembershipFunction ... 3688
A ... 3690
B ... 3690
C ... 3691
GetValue ... 3691
CNormalCombinationMembershipFunction ... 3692
B1 ... 3694
B2 ... 3694
Sigma1 ... 3695
Sigma2 ... 3695
GetValue ... 3696
CNormalMembershipFunction ... 3697
B ... 3699
Sigma ... 3699
GetValue ... 3700
CP_ShapedMembershipFunction ... 3701
A ... 3703
B ... 3703
C ... 3704
D ... 3704
GetValue ... 3705
CProductTwoSigmoidalMembershipFunctions ... 3706
A1 ... 3708
A2 ... 3708
C1 ... 3709
C2 ... 3709
GetValue ... 3710
CS_ShapedMembershipFunction ... 3711
A ... 3713
B ... 3713
GetValue ... 3714
CSigmoidalMembershipFunction ... 3715
A ... 3717
C ... 3717
GetValue ... 3718
CTrapezoidMembershipFunction ... 3719
X1 ... 3721
X2 ... 3721
X3 ... 3722
X4 ... 3722
GetValue ... 3723
CTriangularMembershipFunction ... 3724
X1 ... 3726
X2 ... 3726
X3 ... 3727
ToNormalMF ... 3727
GetValue ... 3727
CZ_ShapedMembershipFunction ... 3729
A ... 3731

© 2000-2025, MetaQuotes Ltd.

B ... 3731
GetValue ... 3732
IMembershipFunction ... 3733
GetValue ... 3733
Правила для нечетких систем ... 3734
CMamdaniFuzzyRule ... 3735
Conclusion ... 3736
Weight ... 3736
CSugenoFuzzyRule ... 3737
Conclusion ... 3738
CSingleCondition ... 3739
Not ... 3739
Term ... 3740
Var ... 3740
CConditions ... 3742
ConditionsList ... 3742
Not ... 3743
Op ... 3743
CGenericFuzzyRule ... 3745
Conclusion ... 3745
Condition ... 3746
CreateCondition ... 3746
Переменные для нечетких систем ... 3748
CFuzzyVariable ... 3749
AddTerm ... 3750
GetTermByName ... 3750
Max ... 3750
Min ... 3751
Terms ... 3751
Values ... 3752
CSugenoVariable ... 3753
Functions ... 3753
GetFuncByName ... 3754
Values ... 3754
Термы нечетких переменных ... 3755
MembershipFunction ... 3756
Нечеткие системы ... 3757
Система Мамдани ... 3758
AggregationMethod ... 3758
Calculate ... 3759
DefuzzificationMethod ... 3759
EmptyRule ... 3759
ImplicationMethod ... 3759
Output ... 3760
OutputByName ... 3760
ParseRule ... 3760
Rules ... 3761
Система Сугено ... 3762
Calculate ... 3762
CreateSugenoFunction ... 3763
EmptyRule ... 3764
Output ... 3764
OutputByName ... 3764
ParseRule ... 3764
Rules ... 3765
OpenCL ... 3766
BufferCreate ... 3768
BufferFree ... 3769

© 2000-2025, MetaQuotes Ltd.

BufferFromArray ... 3770
BufferRead ... 3771
BufferWrite ... 3772
Execute ... 3773
GetContext ... 3774
GetKernel ... 3775
GetKernelName ... 3776
GetProgram ... 3777
Initialize ... 3778
KernelCreate ... 3779
KernelFree ... 3780
SetArgument ... 3781
SetArgumentBuffer ... 3782
SetArgumentLocalMemory ... 3783
SetBuffersCount ... 3784
SetKernelsCount ... 3785
Shutdown ... 3786
SupportDouble ... 3787
Базовый класс CObject ... 3788
Prev ... 3789
Prev ... 3790
Next ... 3791
Next ... 3792
Compare ... 3793
Save ... 3795
Load ... 3797
Type ... 3799
Коллекции данных ... 3800
CArray ... 3802
Step ... 3804
Step ... 3805
Total ... 3806
Available ... 3807
Max ... 3808
IsSorted ... 3809
SortMode ... 3810
Clear ... 3811
Sort ... 3812
Save ... 3813
Load ... 3815
CArrayChar ... 3817
Reserve ... 3820
Resize ... 3821
Shutdown ... 3822
Add ... 3823
AddArray ... 3824
AddArray ... 3825
Insert ... 3827
InsertArray ... 3828
InsertArray ... 3830
AssignArray ... 3832
AssignArray ... 3833
Update ... 3835
Shift ... 3836
Delete ... 3837
DeleteRange ... 3838
At ... 3839
CompareArray ... 3841

© 2000-2025, MetaQuotes Ltd.

CompareArray ... 3842
InsertSort ... 3843
Search ... 3844
SearchGreat ... 3845
SearchLess ... 3846
SearchGreatOrEqual ... 3847
SearchLessOrEqual ... 3848
SearchFirst ... 3849
SearchLast ... 3850
SearchLinear ... 3851
Save ... 3852
Load ... 3854
Type ... 3856
CArrayShort ... 3857
Reserve ... 3860
Resize ... 3861
Shutdown ... 3862
Add ... 3863
AddArray ... 3864
AddArray ... 3865
Insert ... 3867
InsertArray ... 3868
InsertArray ... 3869
AssignArray ... 3871
AssignArray ... 3872
Update ... 3874
Shift ... 3875
Delete ... 3876
DeleteRange ... 3877
At ... 3878
CompareArray ... 3880
CompareArray ... 3881
InsertSort ... 3882
Search ... 3883
SearchGreat ... 3884
SearchLess ... 3885
SearchGreatOrEqual ... 3886
SearchLessOrEqual ... 3887
SearchFirst ... 3888
SearchLast ... 3889
SearchLinear ... 3890
Save ... 3891
Load ... 3893
Type ... 3895
CArrayInt ... 3896
Reserve ... 3899
Resize ... 3900
Shutdown ... 3901
Add ... 3902
AddArray ... 3903
AddArray ... 3904
Insert ... 3906
InsertArray ... 3907
InsertArray ... 3908
AssignArray ... 3910
AssignArray ... 3911
Update ... 3913
Shift ... 3914

© 2000-2025, MetaQuotes Ltd.

Delete ... 3915
DeleteRange ... 3916
At ... 3917
CompareArray ... 3919
CompareArray ... 3920
InsertSort ... 3921
Search ... 3922
SearchGreat ... 3923
SearchLess ... 3924
SearchGreatOrEqual ... 3925
SearchLessOrEqual ... 3926
SearchFirst ... 3927
SearchLast ... 3928
SearchLinear ... 3929
Save ... 3930
Load ... 3932
Type ... 3934
CArrayLong ... 3935
Reserve ... 3938
Resize ... 3939
Shutdown ... 3940
Add ... 3941
AddArray ... 3942
AddArray ... 3943
Insert ... 3945
InsertArray ... 3946
InsertArray ... 3947
AssignArray ... 3949
AssignArray ... 3950
Update ... 3952
Shift ... 3953
Delete ... 3954
DeleteRange ... 3955
At ... 3956
CompareArray ... 3958
CompareArray ... 3959
InsertSort ... 3960
Search ... 3961
SearchGreat ... 3962
SearchLess ... 3963
SearchGreatOrEqual ... 3964
SearchLessOrEqual ... 3965
SearchFirst ... 3966
SearchLast ... 3967
SearchLinear ... 3968
Save ... 3969
Load ... 3971
Type ... 3973
CArrayFloat ... 3974
Delta ... 3977
Reserve ... 3978
Resize ... 3979
Shutdown ... 3980
Add ... 3981
AddArray ... 3982
AddArray ... 3983
Insert ... 3985
InsertArray ... 3986

© 2000-2025, MetaQuotes Ltd.

InsertArray 3987
AssignArray 3989
AssignArray 3990
Update 3992
Shift 3993
Delete 3994
DeleteRange 3995
At 3996
CompareArray 3998
CompareArray 3999
InsertSort 4000
Search 4001
SearchGreat 4002
SearchLess 4003
SearchGreatOrEqual 4004
SearchLessOrEqual 4005
SearchFirst 4006
SearchLast 4007
SearchLinear 4008
Save 4009
Load 4011
Type 4013
CArrayDouble 4014
Delta 4017
Reserve 4018
Resize 4019
Shutdown 4020
Add 4021
AddArray 4022
AddArray 4023
Insert 4025
InsertArray 4026
InsertArray 4027
AssignArray 4029
AssignArray 4030
Update 4032
Shift 4033
Delete 4034
DeleteRange 4035
At 4036
CompareArray 4038
CompareArray 4039
Minimum 4040
Maximum 4041
InsertSort 4042
Search 4043
SearchGreat 4044
SearchLess 4045
SearchGreatOrEqual 4046
SearchLessOrEqual 4047
SearchFirst 4048
SearchLast 4049
SearchLinear 4050
Save 4051
Load 4053
Type 4055
CArrayString 4056
Reserve 4059

© 2000-2025, MetaQuotes Ltd.

Resize ... 4060
Shutdown ... 4061
Add ... 4062
AddArray ... 4063
AddArray ... 4064
Insert ... 4066
InsertArray ... 4067
InsertArray ... 4068
AssignArray ... 4070
AssignArray ... 4071
Update ... 4073
Shift ... 4074
Delete ... 4075
DeleteRange ... 4076
At ... 4077
CompareArray ... 4079
CompareArray ... 4080
InsertSort ... 4081
Search ... 4082
SearchGreat ... 4083
SearchLess ... 4084
SearchGreatOrEqual ... 4085
SearchLessOrEqual ... 4086
SearchFirst ... 4087
SearchLast ... 4088
SearchLinear ... 4089
Save ... 4090
Load ... 4092
Type ... 4094
CArrayObj ... 4095
FreeMode ... 4100
FreeMode ... 4101
Reserve ... 4103
Resize ... 4104
Clear ... 4106
Shutdown ... 4107
CreateElement ... 4108
Add ... 4110
AddArray ... 4112
Insert ... 4115
InsertArray ... 4117
AssignArray ... 4119
Update ... 4121
Shift ... 4123
Detach ... 4124
Delete ... 4126
DeleteRange ... 4127
At ... 4129
CompareArray ... 4131
InsertSort ... 4132
Search ... 4133
SearchGreat ... 4135
SearchLess ... 4137
SearchGreatOrEqual ... 4139
SearchLessOrEqual ... 4141
SearchFirst ... 4143
SearchLast ... 4145
Save ... 4147

© 2000-2025, MetaQuotes Ltd.

Load ... 4149
Type ... 4151
CList ... 4152
FreeMode ... 4154
FreeMode ... 4155
Total ... 4157
IsSorted ... 4158
SortMode ... 4159
CreateElement ... 4160
Add ... 4161
Insert ... 4163
DetachCurrent ... 4165
DeleteCurrent ... 4166
Delete ... 4167
Clear ... 4168
IndexOf ... 4169
GetNodeAtIndex ... 4170
GetFirstNode ... 4171
GetPrevNode ... 4172
GetCurrentNode ... 4173
GetNextNode ... 4174
GetLastNode ... 4175
Sort ... 4176
MoveToIndex ... 4177
Exchange ... 4178
CompareList ... 4179
Search ... 4180
Save ... 4182
Load ... 4184
Type ... 4186
CTreeNode ... 4187
Owner ... 4192
Left ... 4193
Right ... 4194
Balance ... 4195
BalanceL ... 4196
BalanceR ... 4197
CreateSample ... 4198
RefreshBalance ... 4199
GetNext ... 4200
SaveNode ... 4201
LoadNode ... 4202
Type ... 4203
CTree ... 4204
Root ... 4210
CreateElement ... 4211
Insert ... 4212
Detach ... 4213
Delete ... 4214
Clear ... 4215
Find ... 4216
Save ... 4217
Load ... 4218
Type ... 4219
Шаблонные коллекции данных ... 4220
ICollection<T> ... 4223
Add ... 4224
Count ... 4225

© 2000-2025, MetaQuotes Ltd.

Contains ... 4226
CopyTo ... 4227
Clear ... 4228
Remove ... 4229
IEqualityComparable<T> ... 4230
Equals ... 4231
HashCode ... 4232
IComparable<T> ... 4233
Compare ... 4234
IComparer<T> ... 4235
Compare ... 4236
IEqualityComparer<T> ... 4237
Equals ... 4238
HashCode ... 4239
IList<T> ... 4240
TryGetValue ... 4241
TrySetValue ... 4242
Insert ... 4243
IndexOf ... 4244
LastIndexOf ... 4245
RemoveAt ... 4246
IMap<TKey,TValue> ... 4247
Add ... 4248
Contains ... 4249
Remove ... 4250
TryGetValue ... 4251
TrySetValue ... 4252
CopyTo ... 4253
ISet<T> ... 4254
ExceptWith ... 4256
IntersectWith ... 4257
SymmetricExceptWith ... 4258
UnionWith ... 4259
IsProperSubsetOf ... 4260
IsProperSupersetOf ... 4261
IsSubsetOf ... 4262
IsSupersetOf ... 4263
Overlaps ... 4264
SetEquals ... 4265
CDefaultComparer<T> ... 4266
Compare ... 4267
CDefaultEqualityComparer<T> ... 4268
Equals ... 4269
HashCode ... 4270
CRedBlackTreeNode<T> ... 4271
Value ... 4272
Parent ... 4273
Left ... 4274
Right ... 4275
Color ... 4276
IsLeaf ... 4277
CreateEmptyNode ... 4278
CLinkedListNode<T> ... 4279
List ... 4280
Next ... 4281
Previous ... 4282
Value ... 4283
CKeyValuePair<TKey,TValue> ... 4284

© 2000-2025, MetaQuotes Ltd.

Key ... 4285
Value ... 4286
Clone ... 4287
Compare ... 4288
Equals ... 4289
HashCode ... 4290
CArrayList<T> ... 4291
Capacity ... 4293
Count ... 4294
Contains ... 4295
TrimExcess ... 4296
TryGetValue ... 4297
TrySetValue ... 4298
Add ... 4299
AddRange ... 4300
Insert ... 4301
InsertRange ... 4302
CopyTo ... 4303
BinarySearch ... 4304
IndexOf ... 4305
LastIndexOf ... 4306
Clear ... 4307
Remove ... 4308
RemoveAt ... 4309
RemoveRange ... 4310
Reverse ... 4311
Sort ... 4312
CHashMap<TKey,TValue> ... 4313
Add ... 4315
Count ... 4316
Comparer ... 4317
Contains ... 4318
ContainsKey ... 4319
ContainsValue ... 4320
CopyTo ... 4321
Clear ... 4322
Remove ... 4323
TryGetValue ... 4324
TrySetValue ... 4325
CHashSet<T> ... 4326
Add ... 4328
Count ... 4329
Contains ... 4330
Comparer ... 4331
TrimExcess ... 4332
CopyTo ... 4333
Clear ... 4334
Remove ... 4335
ExceptWith ... 4336
IntersectWith ... 4337
SymmetricExceptWith ... 4338
UnionWith ... 4339
IsProperSubsetOf ... 4340
IsProperSupersetOf ... 4341
IsSubsetOf ... 4342
IsSupersetOf ... 4343
Overlaps ... 4344
SetEquals ... 4345

© 2000-2025, MetaQuotes Ltd.

CLinkedList<T> 4346
Add 4348
AddAfter 4349
AddBefore 4350
AddFirst 4351
AddLast 4352
Count 4353
Head 4354
First 4355
Last 4356
Contains 4357
CopyTo 4358
Clear 4359
Remove 4360
RemoveFirst 4361
RemoveLast 4362
Find 4363
FindLast 4364
CQueue<T> 4365
Add 4366
Enqueue 4367
Count 4368
Contains 4369
TrimExcess 4370
CopyTo 4371
Clear 4372
Remove 4373
Dequeue 4374
Peek 4375
CRedBlackTree<T> 4376
Add 4378
Count 4379
Root 4380
Contains 4381
Comparer 4382
TryGetMin 4383
TryGetMax 4384
CopyTo 4385
Clear 4386
Remove 4387
RemoveMin 4388
RemoveMax 4389
Find 4390
FindMin 4391
FindMax 4392
CSortedMap<TKey,TValue> 4393
Add 4395
Count 4396
Comparer 4397
Contains 4398
ContainsKey 4399
ContainsValue 4400
CopyTo 4401
Clear 4402
Remove 4403
TryGetValue 4404
TrySetValue 4405
CSortedSet<T> 4406

© 2000-2025, MetaQuotes Ltd.

Add ... 4408
Count ... 4409
Contains ... 4410
Comparer ... 4411
TryGetMin ... 4412
TryGetMax ... 4413
CopyTo ... 4414
Clear ... 4415
Remove ... 4416
ExceptWith ... 4417
IntersectWith ... 4418
SymmetricExceptWith ... 4419
UnionWith ... 4420
IsProperSubsetOf ... 4421
IsProperSupersetOf ... 4422
IsSubsetOf ... 4423
IsSupersetOf ... 4424
Overlaps ... 4425
SetEquals ... 4426
GetViewBetween ... 4427
GetReverse ... 4428
CStack<T> ... 4429
Add ... 4430
Count ... 4431
Contains ... 4432
TrimExcess ... 4433
CopyTo ... 4434
Clear ... 4435
Remove ... 4436
Push ... 4437
Peek ... 4438
Pop ... 4439
ArrayBinarySearch<T> ... 4440
ArrayIndexOf<T> ... 4441
ArrayLastIndexOf<T> ... 4442
ArrayReverse<T> ... 4443
Compare ... 4444
Equals<T> ... 4447
GetHashCode ... 4448
Файлы ... 4451
CFile ... 4452
Handle ... 4454
Filename ... 4455
Flags ... 4456
SetUnicode ... 4457
SetCommon ... 4458
Open ... 4459
Close ... 4460
Delete ... 4461
IsExist ... 4462
Copy ... 4463
Move ... 4464
Size ... 4465
Tell ... 4466
Seek ... 4467
Flush ... 4468
IsEnding ... 4469
IsLineEnding ... 4470

© 2000-2025, MetaQuotes Ltd.

FolderCreate ... 4471
FolderDelete ... 4472
FolderClean ... 4473
FileFindFirst ... 4474
FileFindNext ... 4475
FileFindClose ... 4476
CFileBin ... 4477
Open ... 4479
WriteChar ... 4480
WriteShort ... 4481
WriteInteger ... 4482
WriteLong ... 4483
WriteFloat ... 4484
WriteDouble ... 4485
WriteString ... 4486
WriteCharArray ... 4487
WriteShortArray ... 4488
WriteIntegerArray ... 4489
WriteLongArray ... 4490
WriteFloatArray ... 4491
WriteDoubleArray ... 4492
WriteObject ... 4493
ReadChar ... 4494
ReadShort ... 4495
ReadInteger ... 4496
ReadLong ... 4497
ReadFloat ... 4498
ReadDouble ... 4499
ReadString ... 4500
ReadCharArray ... 4501
ReadShortArray ... 4502
ReadIntegerArray ... 4503
ReadLongArray ... 4504
ReadFloatArray ... 4505
ReadDoubleArray ... 4506
ReadObject ... 4507
CFileTxt ... 4508
Open ... 4509
WriteString ... 4510
ReadString ... 4511
Строки ... 4512
CString ... 4513
Str ... 4515
Len ... 4516
Copy ... 4517
Fill ... 4518
Assign ... 4519
Append ... 4520
Insert ... 4521
Compare ... 4522
CompareNoCase ... 4523
Left ... 4524
Right ... 4525
Mid ... 4526
Trim ... 4527
TrimLeft ... 4528
TrimRight ... 4529
Clear ... 4530

© 2000-2025, MetaQuotes Ltd.

ToUpper ... 4531
ToLower ... 4532
Reverse ... 4533
Find ... 4534
FindRev ... 4535
Remove ... 4536
Replace ... 4537
Графические объекты ... 4538
Базовый класс CChartObject ... 4539
ChartId ... 4542
Window ... 4543
Name ... 4544
NumPoints ... 4545
Attach ... 4546
SetPoint ... 4547
Delete ... 4548
Detach ... 4549
ShiftObject ... 4550
ShiftPoint ... 4551
Time ... 4552
Price ... 4554
Color ... 4556
Style ... 4557
Width ... 4558
Background ... 4559
Selected ... 4560
Selectable ... 4561
Description ... 4562
Tooltip ... 4563
Timeframes ... 4564
Z_Order ... 4565
CreateTime ... 4566
LevelsCount ... 4567
LevelColor ... 4568
LevelStyle ... 4570
LevelWidth ... 4572
LevelValue ... 4574
LevelDescription ... 4576
GetInteger ... 4578
SetInteger ... 4580
GetDouble ... 4582
SetDouble ... 4584
GetString ... 4586
SetString ... 4588
Save ... 4590
Load ... 4591
Type ... 4592
Объекты "Линии" ... 4593
CChartObjectVLine ... 4594
Create ... 4595
Type ... 4596
CChartObjectHLine ... 4597
Create ... 4598
Type ... 4599
CChartObjectTrend ... 4600
Create ... 4602
RayLeft ... 4603
RayRight ... 4604

© 2000-2025, MetaQuotes Ltd.

Save ... 4605
Load ... 4606
Type ... 4607
CChartObjectTrendByAngle ... 4608
Create ... 4610
Angle ... 4611
Type ... 4612
CChartObjectCycles ... 4613
Create ... 4614
Type ... 4615
Объекты "Каналы" ... 4616
CChartObjectChannel ... 4617
Create ... 4619
Type ... 4620
CChartObjectRegression ... 4621
Create ... 4623
Type ... 4624
CChartObjectStdDevChannel ... 4625
Create ... 4627
Deviations ... 4628
Save ... 4629
Load ... 4630
Type ... 4631
CChartObjectPitchfork ... 4632
Create ... 4634
Type ... 4635
Инструменты Ганна ... 4636
CChartObjectGannLine ... 4637
Create ... 4639
PipsPerBar ... 4640
Save ... 4641
Load ... 4642
Type ... 4643
CChartObjectGannFan ... 4644
Create ... 4646
PipsPerBar ... 4647
Downtrend ... 4648
Save ... 4649
Load ... 4650
Type ... 4651
CChartObjectGannGrid ... 4652
Create ... 4654
PipsPerBar ... 4655
Downtrend ... 4656
Save ... 4657
Load ... 4658
Type ... 4659
Инструменты Фибоначчи ... 4660
CChartObjectFibo ... 4661
Create ... 4663
Type ... 4664
CChartObjectFiboTimes ... 4665
Create ... 4666
Type ... 4667
CChartObjectFiboFan ... 4668
Create ... 4669
Type ... 4670
CChartObjectFiboArc ... 4671

© 2000-2025, MetaQuotes Ltd.

Create ... 4673
Scale ... 4674
Ellipse ... 4675
Save ... 4676
Load ... 4677
Type ... 4678
CChartObjectFiboChannel ... 4679
Create ... 4681
Type ... 4682
CChartObjectFiboExpansion ... 4683
Create ... 4685
Type ... 4686
Инструменты Эллиотта ... 4687
CChartObjectElliottWave3 ... 4688
Create ... 4690
Degree ... 4691
Lines ... 4692
Save ... 4693
Load ... 4694
Type ... 4695
CChartObjectElliottWave5 ... 4696
Create ... 4698
Type ... 4700
Объекты "Фигуры" ... 4701
CChartObjectRectangle ... 4702
Create ... 4703
Type ... 4704
CChartObjectTriangle ... 4705
Create ... 4706
Type ... 4707
CChartObjectEllipse ... 4708
Create ... 4709
Type ... 4710
Объекты "Стрелки" ... 4711
CChartObjectArrow ... 4712
Create ... 4714
ArrowCode ... 4716
Anchor ... 4718
Save ... 4720
Load ... 4721
Type ... 4722
Стрелки с фиксированным кодом ... 4723
Create ... 4725
ArrowCode ... 4727
Type ... 4729
Элементы управления ... 4730
CChartObjectText ... 4731
Create ... 4733
Angle ... 4734
Font ... 4735
FontSize ... 4736
Anchor ... 4737
Save ... 4738
Load ... 4739
Type ... 4740
CChartObjectLabel ... 4741
Create ... 4743
X_Distance ... 4744

© 2000-2025, MetaQuotes Ltd.

Y_Distance ... 4745
X_Size ... 4746
Y_Size ... 4747
Corner ... 4748
Time ... 4749
Price ... 4750
Save ... 4751
Load ... 4752
Type ... 4753
CChartObjectEdit ... 4754
Create ... 4756
TextAlign ... 4757
X_Size ... 4758
Y_Size ... 4759
BackColor ... 4760
BorderColor ... 4761
ReadOnly ... 4762
Angle ... 4763
Save ... 4764
Load ... 4765
Type ... 4766
CChartObjectButton ... 4767
State ... 4769
Save ... 4770
Load ... 4771
Type ... 4772
CChartObjectSubChart ... 4773
Create ... 4775
X_Distance ... 4776
Y_Distance ... 4777
Corner ... 4778
X_Size ... 4779
Y_Size ... 4780
Symbol ... 4781
Period ... 4782
Scale ... 4783
DateScale ... 4784
PriceScale ... 4785
Time ... 4786
Price ... 4787
Save ... 4788
Load ... 4789
Type ... 4790
CChartObjectBitmap ... 4791
Create ... 4793
BmpFile ... 4794
X_Offset ... 4795
Y_Offset ... 4796
Save ... 4797
Load ... 4798
Type ... 4799
CChartObjectBmpLabel ... 4800
Create ... 4802
X_Distance ... 4803
Y_Distance ... 4804
X_Offset ... 4805
Y_Offset ... 4806
Corner ... 4807

© 2000-2025, MetaQuotes Ltd.

X_Size ... 4808
Y_Size ... 4809
BmpFileOn ... 4810
BmpFileOff ... 4811
State ... 4812
Time ... 4813
Price ... 4814
Save ... 4815
Load ... 4816
Type ... 4817
CChartObjectRectLabel ... 4818
Create ... 4820
X_Size ... 4821
Y_Size ... 4822
BackColor ... 4823
Angle ... 4824
BorderType ... 4825
Save ... 4826
Load ... 4827
Type ... 4828
Пользовательская графика ... 4829
CCanvas ... 4830
Attach ... 4834
Arc ... 4835
Pie ... 4838
FillPolygon ... 4842
FillEllipse ... 4843
GetDefaultColor ... 4844
ChartObjectName ... 4845
Circle ... 4846
CircleAA ... 4847
CircleWu ... 4848
Create ... 4849
CreateBitmap ... 4850
CreateBitmapLabel ... 4852
Destroy ... 4854
Ellipse ... 4855
EllipseAA ... 4856
EllipseWu ... 4857
Erase ... 4858
Fill ... 4859
FillCircle ... 4860
FillRectangle ... 4861
FillTriangle ... 4862
FontAngleGet ... 4863
FontAngleSet ... 4864
FontFlagsGet ... 4865
FontFlagsSet ... 4866
FontGet ... 4867
FontNameGet ... 4868
FontNameSet ... 4869
FontSet ... 4870
FontSizeGet ... 4871
FontSizeSet ... 4872
Height ... 4873
Line ... 4874
LineAA ... 4875
LineWu ... 4876

© 2000-2025, MetaQuotes Ltd.

LineHorizontal ... 4877
LineVertical ... 4878
LineStyleSet ... 4879
LineThick ... 4880
LineThickVertical ... 4882
LineThickHorizontal ... 4883
LoadFromFile ... 4884
PixelGet ... 4885
PixelSet ... 4886
PixelSetAA ... 4887
Polygon ... 4888
PolygonAA ... 4889
PolygonWu ... 4890
PolygonThick ... 4891
PolygonSmooth ... 4892
Polyline ... 4893
PolylineSmooth ... 4894
PolylineThick ... 4895
PolylineWu ... 4896
PolylineAA ... 4897
Rectangle ... 4898
Resize ... 4899
ResourceName ... 4900
TextHeight ... 4901
TextOut ... 4902
TextSize ... 4903
TextWidth ... 4904
TransparentLevelSet ... 4905
Triangle ... 4906
TriangleAA ... 4907
TriangleWu ... 4908
Update ... 4909
Width ... 4910
CChartCanvas ... 4911
ColorBackground ... 4915
ColorBorder ... 4916
ColorText ... 4917
ColorGrid ... 4918
MaxData ... 4919
MaxDescrLen ... 4920
ShowFlags ... 4921
IsShowLegend ... 4922
IsShowScaleLeft ... 4923
IsShowScaleRight ... 4924
IsShowScaleTop ... 4925
IsShowScaleBottom ... 4926
IsShowGrid ... 4927
IsShowDescriptors ... 4928
IsShowPercent ... 4929
VScaleMin ... 4930
VScaleMax ... 4931
NumGrid ... 4932
DataOffset ... 4933
DataTotal ... 4934
DrawDescriptors ... 4935
DrawData ... 4936
Create ... 4937
AllowedShowFlags ... 4938

© 2000-2025, MetaQuotes Ltd.

ShowLegend ... 4939
ShowScaleLeft ... 4940
ShowScaleRight ... 4941
ShowScaleTop ... 4942
ShowScaleBottom ... 4943
ShowGrid ... 4944
ShowDescriptors ... 4945
ShowValue ... 4946
ShowPercent ... 4947
LegendAlignment ... 4948
Accumulative ... 4949
VScaleParams ... 4950
DescriptorUpdate ... 4951
ColorUpdate ... 4952
ValuesCheck ... 4953
Redraw ... 4954
DrawBackground ... 4955
DrawLegend ... 4956
DrawLegendVertical ... 4957
DrawLegendHorizontal ... 4958
CalcScales ... 4959
DrawScales ... 4960
DrawScaleLeft ... 4961
DrawScaleRight ... 4962
DrawScaleTop ... 4963
DrawScaleBottom ... 4964
DrawGrid ... 4965
DrawChart ... 4966
CHistogramChart ... 4967
Gradient ... 4972
BarGap ... 4973
BarMinSize ... 4974
BarBorder ... 4975
Create ... 4976
SeriesAdd ... 4977
SeriesInsert ... 4978
SeriesUpdate ... 4979
SeriesDelete ... 4980
ValueUpdate ... 4981
DrawData ... 4982
DrawBar ... 4983
GradientBrush ... 4984
CLineChart ... 4985
Filled ... 4989
Create ... 4990
SeriesAdd ... 4991
SeriesInsert ... 4992
SeriesUpdate ... 4993
SeriesDelete ... 4994
ValueUpdate ... 4995
DrawChart ... 4996
DrawData ... 4997
CalcArea ... 4998
CPieChart ... 4999
Create ... 5004
SeriesSet ... 5005
ValueAdd ... 5006
ValueInsert ... 5007

© 2000-2025, MetaQuotes Ltd.

ValueUpdate 5008
ValueDelete 5009
DrawChart 5010
DrawPie 5011
LabelMake 5012
3D графика 5013
CCanvas3D 5014
AmbientColorGet 5016
AmbientColorSet 5017
Attach 5018
Create 5019
Destroy 5020
DXContext 5021
DXDispatcher 5022
InputScene 5023
LightColorGet 5024
LightColorSet 5025
LightDirectionGet 5026
LightDirectionSet 5027
ObjectAdd 5028
ProjectionMatrixGet 5029
ProjectionMatrixSet 5030
Render 5031
RenderBegin 5032
RenderEnd 5033
ViewMatrixGet 5034
ViewMatrixSet 5035
ViewPositionSet 5036
ViewRotationSet 5037
ViewTargetSet 5038
ViewUpDirectionSet 5039
Ценовые графики 5040
ChartID 5045
Mode 5046
Foreground 5047
Shift 5048
ShiftSize 5049
AutoScroll 5050
Scale 5051
ScaleFix 5052
ScaleFix_11 5053
FixedMax 5054
FixedMin 5055
ScalePPB 5056
PointsPerBar 5057
ShowOHLC 5058
ShowLineBid 5059
ShowLineAsk 5060
ShowLastLine 5061
ShowPeriodSep 5062
ShowGrid 5063
ShowVolumes 5064
ShowObjectDescr 5065
ShowDateScale 5066
ShowPriceScale 5067
ColorBackground 5068
ColorForeground 5069
ColorGrid 5070

© 2000-2025, MetaQuotes Ltd.

ColorBarUp ... 5071
ColorBarDown ... 5072
ColorCandleBull ... 5073
ColorCandleBear ... 5074
ColorChartLine ... 5075
ColorVolumes ... 5076
ColorLineBid ... 5077
ColorLineAsk ... 5078
ColorLineLast ... 5079
ColorStopLevels ... 5080
VisibleBars ... 5081
WindowsTotal ... 5082
WindowIsVisible ... 5083
WindowHandle ... 5084
FirstVisibleBar ... 5085
WidthInBars ... 5086
WidthInPixels ... 5087
HeightInPixels ... 5088
PriceMin ... 5089
PriceMax ... 5090
Attach ... 5091
FirstChart ... 5092
NextChart ... 5093
Open ... 5094
Detach ... 5095
Close ... 5096
BringToTop ... 5097
EventObjectCreate ... 5098
EventObjectDelete ... 5099
IndicatorAdd ... 5100
IndicatorDelete ... 5101
IndicatorsTotal ... 5102
IndicatorName ... 5103
Navigate ... 5104
Symbol ... 5105
Period ... 5106
Redraw ... 5107
GetInteger ... 5108
SetInteger ... 5109
GetDouble ... 5110
SetDouble ... 5111
GetString ... 5112
SetString ... 5113
SetSymbolPeriod ... 5114
ApplyTemplate ... 5115
ScreenShot ... 5116
WindowOnDropped ... 5117
PriceOnDropped ... 5118
TimeOnDropped ... 5119
XOnDropped ... 5120
YOnDropped ... 5121
Save ... 5122
Load ... 5123
Type ... 5124
Научные графики ... 5125
GraphPlot ... 5126
CAxis ... 5130
AutoScale ... 5132

© 2000-2025, MetaQuotes Ltd.

Min ... 5133
Max ... 5134
Step ... 5135
Name ... 5136
Color ... 5137
ValuesSize ... 5138
ValuesWidth ... 5139
ValuesFormat ... 5140
ValuesDateTimeMode ... 5141
ValuesFunctionFormat ... 5142
ValuesFunctionFormatCBData ... 5144
NameSize ... 5145
ZeroLever ... 5146
DefaultStep ... 5147
MaxLabels ... 5148
MinGrace ... 5149
MaxGrace ... 5150
SelectAxisScale ... 5151
CColorGenerator ... 5152
Next ... 5153
Reset ... 5154
CCurve ... 5155
Type ... 5158
Name ... 5159
Color ... 5160
XMax ... 5161
XMin ... 5162
YMax ... 5163
YMin ... 5164
Size ... 5165
PointsSize ... 5166
PointsFill ... 5167
PointsColor ... 5168
GetX ... 5169
GetY ... 5170
LinesStyle ... 5171
LinesIsSmooth ... 5172
LinesSmoothTension ... 5173
LinesSmoothStep ... 5174
LinesEndStyle ... 5175
LinesWidth ... 5176
HistogramWidth ... 5178
CustomPlotCBData ... 5179
CustomPlotFunction ... 5180
PointsType ... 5184
StepsDimension ... 5185
TrendLineCoefficients ... 5186
TrendLineColor ... 5187
TrendLineVisible ... 5188
Update ... 5190
Visible ... 5192
CGraphic ... 5193
Create ... 5196
Destroy ... 5197
Update ... 5198
ChartObjectName ... 5199
ResourceName ... 5200
XAxis ... 5201

© 2000-2025, MetaQuotes Ltd.

YAxis ... 5202
GapSize ... 5203
BackgroundColor ... 5204
BackgroundMain ... 5205
BackgroundMainSize ... 5206
BackgroundMainColor ... 5207
BackgroundSub ... 5208
BackgroundSubSize ... 5209
BackgroundSubColor ... 5210
GridLineColor ... 5211
GridBackgroundColor ... 5212
GridCircleRadius ... 5213
GridCircleColor ... 5214
GridHasCircle ... 5215
GridAxisLineColor ... 5216
HistoryNameWidth ... 5217
HistoryNameSize ... 5218
HistorySymbolSize ... 5219
TextAdd ... 5220
LineAdd ... 5221
CurveAdd ... 5222
CurvePlot ... 5225
CurvePlotAll ... 5226
CurveGetByIndex ... 5227
CurveGetByName ... 5228
CurveRemoveByIndex ... 5229
CurveRemoveByName ... 5230
CurvesTotal ... 5231
MarksToAxisAdd ... 5232
MajorMarkSize ... 5233
FontSet ... 5234
FontGet ... 5235
Attach ... 5236
CalculateMaxMinValues ... 5237
Height ... 5238
IndentDown ... 5239
IndentLeft ... 5240
IndentRight ... 5241
IndentUp ... 5242
Redraw ... 5243
ResetParameters ... 5244
ScaleX ... 5245
ScaleY ... 5246
SetDefaultParameters ... 5247
Width ... 5248
Индикаторы ... 5249
Базовые классы ... 5250
CSpreadBuffer ... 5251
Size ... 5253
SetSymbolPeriod ... 5254
At ... 5255
Refresh ... 5256
RefreshCurrent ... 5257
CTimeBuffer ... 5258
Size ... 5260
SetSymbolPeriod ... 5261
At ... 5262
Refresh ... 5263

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent 5264
CTickVolumeBuffer 5265
Size 5267
SetSymbolPeriod 5268
At 5269
Refresh 5270
RefreshCurrent 5271
CRealVolumeBuffer 5272
Size 5274
SetSymbolPeriod 5275
At 5276
Refresh 5277
RefreshCurrent 5278
CDoubleBuffer 5279
Size 5281
SetSymbolPeriod 5282
At 5283
Refresh 5284
RefreshCurrent 5285
COpenBuffer 5286
Refresh 5287
RefreshCurrent 5288
CHighBuffer 5289
Refresh 5290
RefreshCurrent 5291
CLowBuffer 5292
Refresh 5293
RefreshCurrent 5294
CCloseBuffer 5295
Refresh 5296
RefreshCurrent 5297
CIndicatorBuffer 5298
Offset 5300
Name 5301
At 5302
Refresh 5303
RefreshCurrent 5304
CSeries 5305
Name 5307
BuffersTotal 5308
Timeframe 5309
Symbol 5310
Period 5311
RefreshCurrent 5312
BufferSize 5313
BufferResize 5314
Refresh 5315
PeriodDescription 5316
CPriceSeries 5317
BufferResize 5319
GetData 5320
Refresh 5321
MinIndex 5322
MinValue 5323
MaxIndex 5324
MaxValue 5325
CIndicator 5326
Handle 5328

© 2000-2025, MetaQuotes Ltd.

Status ... 5329
FullRelease ... 5330
Create ... 5331
BufferResize ... 5332
BarsCalculated ... 5333
GetData ... 5334
Refresh ... 5337
Minimum ... 5338
MinValue ... 5339
Maximum ... 5340
MaxValue ... 5341
MethodDescription ... 5342
PriceDescription ... 5343
VolumeDescription ... 5344
AddToChart ... 5345
DeleteFromChart ... 5346
CIndicators ... 5347
Create ... 5348
Refresh ... 5349
Таймсерии ... 5350
CiSpread ... 5351
Create ... 5353
BufferResize ... 5354
GetData ... 5355
Refresh ... 5357
CiTime ... 5358
Create ... 5360
BufferResize ... 5361
GetData ... 5362
Refresh ... 5364
CiTickVolume ... 5365
Create ... 5367
BufferResize ... 5368
GetData ... 5369
Refresh ... 5371
CiRealVolume ... 5372
Create ... 5374
BufferResize ... 5375
GetData ... 5376
Refresh ... 5378
CiOpen ... 5379
Create ... 5381
GetData ... 5382
CiHigh ... 5384
Create ... 5386
GetData ... 5387
CiLow ... 5389
Create ... 5391
GetData ... 5392
CiClose ... 5394
Create ... 5396
GetData ... 5397
Индикаторы тренда ... 5399
CiADX ... 5400
MaPeriod ... 5402
Create ... 5403
Main ... 5404
Plus ... 5405

© 2000-2025, MetaQuotes Ltd.

Minus ... 5406
Type ... 5407
CiADXWilder ... 5408
MaPeriod ... 5410
Create ... 5411
Main ... 5412
Plus ... 5413
Minus ... 5414
Type ... 5415
CiBands ... 5416
MaPeriod ... 5418
MaShift ... 5419
Deviation ... 5420
Applied ... 5421
Create ... 5422
Base ... 5423
Upper ... 5424
Lower ... 5425
Type ... 5426
CiEnvelopes ... 5427
MaPeriod ... 5429
MaShift ... 5430
MaMethod ... 5431
Deviation ... 5432
Applied ... 5433
Create ... 5434
Upper ... 5435
Lower ... 5436
Type ... 5437
CiIchimoku ... 5438
TenkanSenPeriod ... 5440
KijunSenPeriod ... 5441
SenkouSpanBPeriod ... 5442
Create ... 5443
TenkanSen ... 5444
KijunSen ... 5445
SenkouSpanA ... 5446
SenkouSpanB ... 5447
ChinkouSpan ... 5448
Type ... 5449
CiMA ... 5450
MaPeriod ... 5452
MaShift ... 5453
MaMethod ... 5454
Applied ... 5455
Create ... 5456
Main ... 5457
Type ... 5458
CiSAR ... 5459
SarStep ... 5461
Maximum ... 5462
Create ... 5463
Main ... 5464
Type ... 5465
CiStdDev ... 5466
MaPeriod ... 5468
MaShift ... 5469
MaMethod ... 5470

© 2000-2025, MetaQuotes Ltd.

Applied ... 5471
Create ... 5472
Main ... 5473
Type ... 5474
CiDEMA ... 5475
MaPeriod ... 5477
IndShift ... 5478
Applied ... 5479
Create ... 5480
Main ... 5481
Type ... 5482
CiTEMA ... 5483
MaPeriod ... 5485
IndShift ... 5486
Applied ... 5487
Create ... 5488
Main ... 5489
Type ... 5490
CiFrAMA ... 5491
MaPeriod ... 5493
IndShift ... 5494
Applied ... 5495
Create ... 5496
Main ... 5497
Type ... 5498
CiAMA ... 5499
MaPeriod ... 5501
FastEmaPeriod ... 5502
SlowEmaPeriod ... 5503
IndShift ... 5504
Applied ... 5505
Create ... 5506
Main ... 5507
Type ... 5508
CiVIDyA ... 5509
CmoPeriod ... 5511
EmaPeriod ... 5512
IndShift ... 5513
Applied ... 5514
Create ... 5515
Main ... 5516
Type ... 5517
Осцилляторы ... 5518
CiATR ... 5519
MaPeriod ... 5521
Create ... 5522
Main ... 5523
Type ... 5524
CiBearsPower ... 5525
MaPeriod ... 5527
Create ... 5528
Main ... 5529
Type ... 5530
CiBullsPower ... 5531
MaPeriod ... 5533
Create ... 5534
Main ... 5535
Type ... 5536

© 2000-2025, MetaQuotes Ltd.

CiCCI ... 5537
MaPeriod ... 5539
Applied ... 5540
Create ... 5541
Main ... 5542
Type ... 5543
CiChaikin ... 5544
FastMaPeriod ... 5546
SlowMaPeriod ... 5547
MaMethod ... 5548
Applied ... 5549
Create ... 5550
Main ... 5551
Type ... 5552
CiDeMarker ... 5553
MaPeriod ... 5555
Create ... 5556
Main ... 5557
Type ... 5558
CiForce ... 5559
MaPeriod ... 5561
MaMethod ... 5562
Applied ... 5563
Create ... 5564
Main ... 5565
Type ... 5566
CiMACD ... 5567
FastEmaPeriod ... 5569
SlowEmaPeriod ... 5570
SignalPeriod ... 5571
Applied ... 5572
Create ... 5573
Main ... 5574
Signal ... 5575
Type ... 5576
CiMomentum ... 5577
MaPeriod ... 5579
Applied ... 5580
Create ... 5581
Main ... 5582
Type ... 5583
CiOsMA ... 5584
FastEmaPeriod ... 5586
SlowEmaPeriod ... 5587
SignalPeriod ... 5588
Applied ... 5589
Create ... 5590
Main ... 5591
Type ... 5592
CiRSI ... 5593
MaPeriod ... 5595
Applied ... 5596
Create ... 5597
Main ... 5598
Type ... 5599
CiRVI ... 5600
MaPeriod ... 5602
Create ... 5603

© 2000-2025, MetaQuotes Ltd.

Main ... 5604
Signal ... 5605
Type ... 5606
CiStochastic ... 5607
Kperiod ... 5609
Dperiod ... 5610
Slowing ... 5611
MaMethod ... 5612
PriceField ... 5613
Create ... 5614
Main ... 5615
Signal ... 5616
Type ... 5617
CiTriX ... 5618
MaPeriod ... 5620
Applied ... 5621
Create ... 5622
Main ... 5623
Type ... 5624
CiWPR ... 5625
CalcPeriod ... 5627
Create ... 5628
Main ... 5629
Type ... 5630
Индикаторы объема ... 5631
CiAD ... 5632
Applied ... 5634
Create ... 5635
Main ... 5636
Type ... 5637
CiMFI ... 5638
MaPeriod ... 5640
Applied ... 5641
Create ... 5642
Main ... 5643
Type ... 5644
CiOBV ... 5645
Applied ... 5647
Create ... 5648
Main ... 5649
Type ... 5650
CiVolumes ... 5651
Applied ... 5653
Create ... 5654
Main ... 5655
Type ... 5656
Индикаторы Билла Вильямса ... 5657
CiAC ... 5658
Create ... 5660
Main ... 5661
Type ... 5662
CiAlligator ... 5663
JawPeriod ... 5665
JawShift ... 5666
TeethPeriod ... 5667
TeethShift ... 5668
LipsPeriod ... 5669
LipsShift ... 5670

© 2000-2025, MetaQuotes Ltd.

MaMethod 5671
Applied 5672
Create 5673
Jaw 5674
Teeth 5675
Lips 5676
Type 5677
CiAO 5678
Create 5680
Main 5681
Type 5682
CiFractals 5683
Create 5685
Upper 5686
Lower 5687
Type 5688
CiGator 5689
JawPeriod 5691
JawShift 5692
TeethPeriod 5693
TeethShift 5694
LipsPeriod 5695
LipsShift 5696
MaMethod 5697
Applied 5698
Create 5699
Upper 5701
Lower 5702
Type 5703
CiBWMFI 5704
Applied 5706
Create 5707
Main 5708
Type 5709
Пользовательский индикатор 5710
NumBuffers 5711
NumParams 5712
ParamType 5713
ParamLong 5714
ParamDouble 5715
ParamString 5716
Type 5717
Торговые классы 5718
CAccountInfo 5719
Login 5721
TradeMode 5722
TradeModeDescription 5723
Leverage 5724
StopoutMode 5725
StopoutModeDescription 5726
MarginMode 5727
MarginModeDescription 5728
TradeAllowed 5729
TradeExpert 5730
LimitOrders 5731
Balance 5732
Credit 5733
Profit 5734

© 2000-2025, MetaQuotes Ltd.

Equity ... 5735
Margin ... 5736
FreeMargin ... 5737
MarginLevel ... 5738
MarginCall ... 5739
MarginStopOut ... 5740
Name ... 5741
Server ... 5742
Currency ... 5743
Company ... 5744
InfoInteger ... 5745
InfoDouble ... 5746
InfoString ... 5747
OrderProfitCheck ... 5748
MarginCheck ... 5749
FreeMarginCheck ... 5750
MaxLotCheck ... 5751
CSymbolInfo ... 5752
Refresh ... 5756
RefreshRates ... 5757
Name ... 5758
Select ... 5759
IsSynchronized ... 5760
Volume ... 5761
VolumeHigh ... 5762
VolumeLow ... 5763
Time ... 5764
Spread ... 5765
SpreadFloat ... 5766
TicksBookDepth ... 5767
StopsLevel ... 5768
FreezeLevel ... 5769
Bid ... 5770
BidHigh ... 5771
BidLow ... 5772
Ask ... 5773
AskHigh ... 5774
AskLow ... 5775
Last ... 5776
LastHigh ... 5777
LastLow ... 5778
TradeCalcMode ... 5779
TradeCalcModeDescription ... 5780
TradeMode ... 5781
TradeModeDescription ... 5782
TradeExecution ... 5783
TradeExecutionDescription ... 5784
SwapMode ... 5785
SwapModeDescription ... 5786
SwapRollover3days ... 5787
SwapRollover3daysDescription ... 5788
MarginInitial ... 5789
MarginMaintenance ... 5790
MarginLong ... 5791
MarginShort ... 5792
MarginLimit ... 5793
MarginStop ... 5794
MarginStopLimit ... 5795

© 2000-2025, MetaQuotes Ltd.

TradeTimeFlags ... 5796
TradeFillFlags ... 5797
Digits ... 5798
Point ... 5799
TickValue ... 5800
TickValueProfit ... 5801
TickValueLoss ... 5802
TickSize ... 5803
ContractSize ... 5804
LotsMin ... 5805
LotsMax ... 5806
LotsStep ... 5807
LotsLimit ... 5808
SwapLong ... 5809
SwapShort ... 5810
CurrencyBase ... 5811
CurrencyProfit ... 5812
CurrencyMargin ... 5813
Bank ... 5814
Description ... 5815
Path ... 5816
SessionDeals ... 5817
SessionBuyOrders ... 5818
SessionSellOrders ... 5819
SessionTurnover ... 5820
SessionInterest ... 5821
SessionBuyOrdersVolume ... 5822
SessionSellOrdersVolume ... 5823
SessionOpen ... 5824
SessionClose ... 5825
SessionAW ... 5826
SessionPriceSettlement ... 5827
SessionPriceLimitMin ... 5828
SessionPriceLimitMax ... 5829
InfoInteger ... 5830
InfoDouble ... 5831
InfoString ... 5832
NormalizePrice ... 5833
COrderInfo ... 5834
Ticket ... 5836
TimeSetup ... 5837
TimeSetupMsc ... 5838
OrderType ... 5839
TypeDescription ... 5840
State ... 5841
StateDescription ... 5842
TimeExpiration ... 5843
TimeDone ... 5844
TimeDoneMsc ... 5845
TypeFilling ... 5846
TypeFillingDescription ... 5847
TypeTime ... 5848
TypeTimeDescription ... 5849
Magic ... 5850
PositionId ... 5851
VolumeInitial ... 5852
VolumeCurrent ... 5853
PriceOpen ... 5854

© 2000-2025, MetaQuotes Ltd.

StopLoss ... 5855
TakeProfit ... 5856
PriceCurrent ... 5857
PriceStopLimit ... 5858
Symbol ... 5859
Comment ... 5860
InfoInteger ... 5861
InfoDouble ... 5862
InfoString ... 5863
StoreState ... 5864
CheckState ... 5865
Select ... 5866
SelectByIndex ... 5867
CHistoryOrderInfo ... 5868
TimeSetup ... 5870
TimeSetupMsc ... 5871
OrderType ... 5872
TypeDescription ... 5873
State ... 5874
StateDescription ... 5875
TimeExpiration ... 5876
TimeDone ... 5877
TimeDoneMsc ... 5878
TypeFilling ... 5879
TypeFillingDescription ... 5880
TypeTime ... 5881
TypeTimeDescription ... 5882
Magic ... 5883
PositionId ... 5884
VolumeInitial ... 5885
VolumeCurrent ... 5886
PriceOpen ... 5887
StopLoss ... 5888
TakeProfit ... 5889
PriceCurrent ... 5890
PriceStopLimit ... 5891
Symbol ... 5892
Comment ... 5893
InfoInteger ... 5894
InfoDouble ... 5895
InfoString ... 5896
Ticket ... 5897
SelectByIndex ... 5898
CPositionInfo ... 5899
Time ... 5901
TimeMsc ... 5902
TimeUpdate ... 5903
TimeUpdateMsc ... 5904
PositionType ... 5905
TypeDescription ... 5906
Magic ... 5907
Identifier ... 5908
Volume ... 5909
PriceOpen ... 5910
StopLoss ... 5911
TakeProfit ... 5912
PriceCurrent ... 5913
Commission ... 5914

© 2000-2025, MetaQuotes Ltd.

Swap ... 5915
Profit ... 5916
Symbol ... 5917
Comment ... 5918
InfoInteger ... 5919
InfoDouble ... 5920
InfoString ... 5921
Select ... 5922
SelectByIndex ... 5923
SelectByMagic ... 5924
SelectByTicket ... 5925
StoreState ... 5926
CheckState ... 5927
CDealInfo ... 5928
Order ... 5930
Time ... 5931
TimeMsc ... 5932
DealType ... 5933
TypeDescription ... 5934
Entry ... 5935
EntryDescription ... 5936
Magic ... 5937
PositionId ... 5938
Volume ... 5939
Price ... 5940
Commision ... 5941
Swap ... 5942
Profit ... 5943
Symbol ... 5944
Comment ... 5945
InfoInteger ... 5946
InfoDouble ... 5947
InfoString ... 5948
Ticket ... 5949
SelectByIndex ... 5950
CTrade ... 5951
LogLevel ... 5955
SetExpertMagicNumber ... 5956
SetDeviationInPoints ... 5957
SetTypeFilling ... 5958
SetTypeFillingBySymbol ... 5959
SetAsyncMode ... 5960
SetMarginMode ... 5961
OrderOpen ... 5962
OrderModify ... 5964
OrderDelete ... 5965
PositionOpen ... 5966
PositionModify ... 5967
PositionClose ... 5969
PositionClosePartial ... 5970
PositionCloseBy ... 5972
Buy ... 5973
Sell ... 5974
BuyLimit ... 5975
BuyStop ... 5976
SellLimit ... 5978
SellStop ... 5980
Request ... 5982

© 2000-2025, MetaQuotes Ltd.

RequestAction ... 5983
RequestActionDescription ... 5984
RequestMagic ... 5985
RequestOrder ... 5986
RequestSymbol ... 5987
RequestVolume ... 5988
RequestPrice ... 5989
RequestStopLimit ... 5990
RequestSL ... 5991
RequestTP ... 5992
RequestDeviation ... 5993
RequestType ... 5994
RequestTypeDescription ... 5995
RequestTypeFilling ... 5996
RequestTypeFillingDescription ... 5997
RequestTypeTime ... 5998
RequestTypeTimeDescription ... 5999
RequestExpiration ... 6000
RequestComment ... 6001
RequestPosition ... 6002
RequestPositionBy ... 6003
Result ... 6004
ResultRetcode ... 6005
ResultRetcodeDescription ... 6006
ResultDeal ... 6007
ResultOrder ... 6008
ResultVolume ... 6009
ResultPrice ... 6010
ResultBid ... 6011
ResultAsk ... 6012
ResultComment ... 6013
CheckResult ... 6014
CheckResultRetcode ... 6015
CheckResultRetcodeDescription ... 6016
CheckResultBalance ... 6017
CheckResultEquity ... 6018
CheckResultProfit ... 6019
CheckResultMargin ... 6020
CheckResultMarginFree ... 6021
CheckResultMarginLevel ... 6022
CheckResultComment ... 6023
PrintRequest ... 6024
PrintResult ... 6025
FormatRequest ... 6026
FormatRequestResult ... 6027
CTerminalInfo ... 6028
Build ... 6030
IsConnected ... 6031
IsDLLsAllowed ... 6032
IsTradeAllowed ... 6033
IsEmailEnabled ... 6034
IsFtpEnabled ... 6035
MaxBars ... 6036
CodePage ... 6037
CPUCores ... 6038
MemoryPhysical ... 6039
MemoryTotal ... 6040
MemoryAvailable ... 6041

© 2000-2025, MetaQuotes Ltd.

MemoryUsed ... 6042
IsX64 ... 6043
OpenCLSupport ... 6044
DiskSpace ... 6045
Language ... 6046
Name ... 6047
Company ... 6048
Path ... 6049
DataPath ... 6050
CommonDataPath ... 6051
InfoInteger ... 6052
InfoString ... 6053
Модули стратегий ... 6054
Базовые классы экспертов ... 6057
CExpertBase ... 6058
InitPhase ... 6061
TrendType ... 6062
UsedSeries ... 6063
EveryTick ... 6064
Open ... 6065
High ... 6066
Low ... 6067
Close ... 6068
Spread ... 6069
Time ... 6070
TickVolume ... 6071
RealVolume ... 6072
Init ... 6073
Symbol ... 6074
Period ... 6075
Magic ... 6076
ValidationSettings ... 6077
SetPriceSeries ... 6078
SetOtherSeries ... 6079
InitIndicators ... 6080
InitOpen ... 6081
InitHigh ... 6082
InitLow ... 6083
InitClose ... 6084
InitSpread ... 6085
InitTime ... 6086
InitTickVolume ... 6087
InitRealVolume ... 6088
PriceLevelUnit ... 6089
StartIndex ... 6090
CompareMagic ... 6091
CExpert ... 6092
Init ... 6097
Magic ... 6098
InitSignal ... 6099
InitTrailing ... 6100
InitMoney ... 6101
InitTrade ... 6102
Deinit ... 6103
OnTickProcess ... 6104
OnTradeProcess ... 6105
OnTimerProcess ... 6106
OnChartEventProcess ... 6107

© 2000-2025, MetaQuotes Ltd.

OnBookEventProcess ... 6108
MaxOrders ... 6109
Signal ... 6110
ValidationSettings ... 6111
InitIndicators ... 6112
OnTick ... 6113
OnTrade ... 6114
OnTimer ... 6115
OnChartEvent ... 6116
OnBookEvent ... 6117
InitParameters ... 6118
DeinitTrade ... 6119
DeinitSignal ... 6120
DeinitTrailing ... 6121
DeinitMoney ... 6122
DeinitIndicators ... 6123
Refresh ... 6124
Processing ... 6125
SelectPosition ... 6127
CheckOpen ... 6128
CheckOpenLong ... 6129
CheckOpenShort ... 6130
OpenLong ... 6131
OpenShort ... 6132
CheckReverse ... 6133
CheckReverseLong ... 6134
CheckReverseShort ... 6135
ReverseLong ... 6136
ReverseShort ... 6138
CheckClose ... 6140
CheckCloseLong ... 6141
CheckCloseShort ... 6142
CloseAll ... 6143
Close ... 6144
CloseLong ... 6145
CloseShort ... 6146
CheckTrailingStop ... 6147
CheckTrailingStopLong ... 6148
CheckTrailingStopShort ... 6149
TrailingStopLong ... 6150
TrailingStopShort ... 6151
CheckTrailingOrderLong ... 6152
CheckTrailingOrderShort ... 6153
TrailingOrderLong ... 6154
TrailingOrderShort ... 6155
CheckDeleteOrderLong ... 6156
CheckDeleteOrderShort ... 6157
DeleteOrders ... 6158
DeleteOrder ... 6159
DeleteOrderLong ... 6160
DeleteOrderShort ... 6161
LotOpenLong ... 6162
LotOpenShort ... 6163
LotReverse ... 6164
PrepareHistoryDate ... 6165
HistoryPoint ... 6166
CheckTradeState ... 6167
WaitEvent ... 6168

© 2000-2025, MetaQuotes Ltd.

NoWaitEvent ... 6169
TradeEventPositionStopTake ... 6170
TradeEventOrderTriggered ... 6171
TradeEventPositionOpened ... 6172
TradeEventPositionVolumeChanged ... 6173
TradeEventPositionModified ... 6174
TradeEventPositionClosed ... 6175
TradeEventOrderPlaced ... 6176
TradeEventOrderModified ... 6177
TradeEventOrderDeleted ... 6178
TradeEventNotIdentified ... 6179
TimeframeAdd ... 6180
TimeframesFlags ... 6181
CExpertSignal ... 6182
BasePrice ... 6185
UsedSeries ... 6186
Weight ... 6187
PatternsUsage ... 6188
General ... 6189
Ignore ... 6190
Invert ... 6191
ThresholdOpen ... 6192
ThresholdClose ... 6193
PriceLevel ... 6194
StopLevel ... 6195
TakeLevel ... 6196
Expiration ... 6197
Magic ... 6198
ValidationSettings ... 6199
InitIndicators ... 6200
AddFilter ... 6201
CheckOpenLong ... 6202
CheckOpenShort ... 6203
OpenLongParams ... 6204
OpenShortParams ... 6205
CheckCloseLong ... 6206
CheckCloseShort ... 6207
CloseLongParams ... 6208
CloseShortParams ... 6209
CheckReverseLong ... 6210
CheckReverseShort ... 6211
CheckTrailingOrderLong ... 6212
CheckTrailingOrderShort ... 6213
LongCondition ... 6214
ShortCondition ... 6215
Direction ... 6216
CExpertTrailing ... 6217
CheckTrailingStopLong ... 6219
CheckTrailingStopShort ... 6220
CExpertMoney ... 6221
Percent ... 6223
ValidationSettings ... 6224
CheckOpenLong ... 6225
CheckOpenShort ... 6226
CheckReverse ... 6227
CheckClose ... 6228
Модули торговых сигналов ... 6229
Сигналы индикатора Accelerator Oscillator ... 6232

© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Adaptive Moving Average ... 6235
Сигналы индикатора Awesome Oscillator ... 6239
Сигналы осциллятора Bears Power ... 6243
Сигналы осциллятора Bulls Power ... 6245
Сигналы осциллятора Commodity Channel Index ... 6247
Сигналы осциллятора DeMarker ... 6251
Сигналы индикатора Double Exponential Moving Average ... 6255
Сигналы индикатора Envelopes ... 6259
Сигналы индикатора Fractal Adaptive Moving Average ... 6262
Сигналы внутридневного временного фильтра ... 6266
Сигналы осциллятора MACD ... 6268
Сигналы индикатора Moving Average ... 6274
Сигналы индикатора Parabolic SAR ... 6278
Сигналы осциллятора Relative Strength Index ... 6280
Сигналы осциллятора Relative Vigor Index ... 6286
Сигналы осциллятора Stochastic ... 6288
Сигналы осциллятора Triple Exponential Average ... 6293
Сигналы индикатора Triple Exponential Moving Average ... 6297
Сигналы осциллятора Williams Percent Range ... 6301
Модули Trailing Stop ... 6304
CTrailingFixedPips ... 6305
StopLevel ... 6307
ProfitLevel ... 6308
ValidationSettings ... 6309
CheckTrailingStopLong ... 6310
CheckTrailingStopShort ... 6311
CTrailingMA ... 6312
Period ... 6314
Shift ... 6315
Method ... 6316
Applied ... 6317
InitIndicators ... 6318
ValidationSettings ... 6319
CheckTrailingStopLong ... 6320
CheckTrailingStopShort ... 6321
CTrailingNone ... 6322
CheckTrailingStopLong ... 6323
CheckTrailingStopShort ... 6324
CTrailingPSAR ... 6325
Step ... 6327
Maximum ... 6328
InitIndicators ... 6329
CheckTrailingStopLong ... 6330
CheckTrailingStopShort ... 6331
Модули Money Management ... 6332
CMoneyFixedLot ... 6333
Lots ... 6335
ValidationSettings ... 6336
CheckOpenLong ... 6337
CheckOpenShort ... 6338
CMoneyFixedMargin ... 6339
CheckOpenLong ... 6340
CheckOpenShort ... 6341
CMoneyFixedRisk ... 6342
CheckOpenLong ... 6343
CheckOpenShort ... 6344
CMoneyNone ... 6345
ValidationSettings ... 6346

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong ... 6347
CheckOpenShort ... 6348
CMoneySizeOptimized ... 6349
DecreaseFactor ... 6351
ValidationSettings ... 6352
CheckOpenLong ... 6353
CheckOpenShort ... 6354
Панели и диалоги ... 6355
CRect ... 6357
Left ... 6358
Top ... 6359
Right ... 6360
Bottom ... 6361
Width ... 6362
Height ... 6363
SetBound ... 6364
Move ... 6365
Shift ... 6366
Contains ... 6367
Format ... 6368
CDateTime ... 6369
MonthName ... 6371
ShortMonthName ... 6372
DayName ... 6373
ShortDayName ... 6374
DaysInMonth ... 6375
DateTime ... 6376
Date ... 6377
Time ... 6378
Sec ... 6379
Min ... 6380
Hour ... 6381
Day ... 6382
Mon ... 6383
Year ... 6384
SecDec ... 6385
SecInc ... 6386
MinDec ... 6387
MinInc ... 6388
HourDec ... 6389
HourInc ... 6390
DayDec ... 6391
DayInc ... 6392
MonDec ... 6393
MonInc ... 6394
YearDec ... 6395
YearInc ... 6396
CWnd ... 6397
Create ... 6401
Destroy ... 6402
OnEvent ... 6403
OnMouseEvent ... 6404
Name ... 6405
ControlsTotal ... 6406
Control ... 6407
ControlFind ... 6408
Rect ... 6409
Left ... 6410

© 2000-2025, MetaQuotes Ltd.

Top ... 6411
Right ... 6412
Bottom ... 6413
Width ... 6414
Height ... 6415
Move ... 6416
Shift ... 6417
Resize ... 6418
Contains ... 6419
Alignment ... 6420
Align ... 6421
Id ... 6422
IsEnabled ... 6423
Enable ... 6424
Disable ... 6425
IsVisible ... 6426
Visible ... 6427
Show ... 6428
Hide ... 6429
IsActive ... 6430
Activate ... 6431
Deactivate ... 6432
StateFlags ... 6433
StateFlagsSet ... 6434
StateFlagsReset ... 6435
PropFlags ... 6436
PropFlagsSet ... 6437
PropFlagsReset ... 6438
MouseX ... 6439
MouseY ... 6440
MouseFlags ... 6441
MouseFocusKill ... 6442
OnCreate ... 6443
OnDestroy ... 6444
OnMove ... 6445
OnResize ... 6446
OnEnable ... 6447
OnDisable ... 6448
OnShow ... 6449
OnHide ... 6450
OnActivate ... 6451
OnDeactivate ... 6452
OnClick ... 6453
OnChange ... 6454
OnMouseDown ... 6455
OnMouseUp ... 6456
OnDragStart ... 6457
OnDragProcess ... 6458
OnDragEnd ... 6459
DragObjectCreate ... 6460
DragObjectDestroy ... 6461
CWndObj ... 6462
OnEvent ... 6465
Text ... 6466
Color ... 6467
ColorBackground ... 6468
ColorBorder ... 6469
Font ... 6470

© 2000-2025, MetaQuotes Ltd.

FontSize ... 6471
ZOrder ... 6472
OnObjectCreate ... 6473
OnObjectChange ... 6474
OnObjectDelete ... 6475
OnObjectDrag ... 6476
OnSetText ... 6477
OnSetColor ... 6478
OnSetColorBackground ... 6479
OnSetFont ... 6480
OnSetFontSize ... 6481
OnSetZOrder ... 6482
OnDestroy ... 6483
OnChange ... 6484
CWndContainer ... 6485
Destroy ... 6487
OnEvent ... 6488
OnMouseEvent ... 6489
ControlsTotal ... 6490
Control ... 6491
ControlFind ... 6492
Add ... 6493
Delete ... 6494
Move ... 6495
Shift ... 6496
Id ... 6497
Enable ... 6498
Disable ... 6499
Show ... 6500
Hide ... 6501
MouseFocusKill ... 6502
Save ... 6503
Load ... 6504
OnResize ... 6505
OnActivate ... 6506
OnDeactivate ... 6507
CLabel ... 6508
Create ... 6513
OnSetText ... 6514
OnSetColor ... 6515
OnSetFont ... 6516
OnSetFontSize ... 6517
OnCreate ... 6518
OnShow ... 6519
OnHide ... 6520
OnMove ... 6521
CBmpButton ... 6522
Create ... 6529
Border ... 6530
BmpNames ... 6531
BmpOffName ... 6532
BmpOnName ... 6533
BmpPassiveName ... 6534
BmpActiveName ... 6535
Pressed ... 6536
Locking ... 6537
OnSetZOrder ... 6538
OnCreate ... 6539

© 2000-2025, MetaQuotes Ltd.

OnShow ... 6540
OnHide ... 6541
OnMove ... 6542
OnChange ... 6543
OnActivate ... 6544
OnDeactivate ... 6545
OnMouseDown ... 6546
OnMouseUp ... 6547
CButton ... 6548
Create ... 6555
Pressed ... 6556
Locking ... 6557
OnSetText ... 6558
OnSetColor ... 6559
OnSetColorBackground ... 6560
OnSetColorBorder ... 6561
OnSetFont ... 6562
OnSetFontSize ... 6563
OnCreate ... 6564
OnShow ... 6565
OnHide ... 6566
OnMove ... 6567
OnResize ... 6568
OnMouseDown ... 6569
OnMouseUp ... 6570
CEdit ... 6571
Create ... 6577
ReadOnly ... 6578
TextAlign ... 6579
OnObjectEndEdit ... 6580
OnSetText ... 6581
OnSetColor ... 6582
OnSetColorBackground ... 6583
OnSetColorBorder ... 6584
OnSetFont ... 6585
OnSetFontSize ... 6586
OnSetZOrder ... 6587
OnCreate ... 6588
OnShow ... 6589
OnHide ... 6590
OnMove ... 6591
OnResize ... 6592
OnChange ... 6593
OnClick ... 6594
CPanel ... 6595
Create ... 6600
BorderType ... 6601
OnSetText ... 6602
OnSetColorBackground ... 6603
OnSetColorBorder ... 6604
OnCreate ... 6605
OnShow ... 6606
OnHide ... 6607
OnMove ... 6608
OnResize ... 6609
OnChange ... 6610
CPicture ... 6611
Create ... 6616

© 2000-2025, MetaQuotes Ltd.

Border ... 6617
BmpName ... 6618
OnCreate ... 6619
OnShow ... 6620
OnHide ... 6621
OnMove ... 6622
OnChange ... 6623
CScroll ... 6624
Create ... 6626
OnEvent ... 6627
MinPos ... 6628
MaxPos ... 6629
CurrPos ... 6630
CreateBack ... 6631
CreateInc ... 6632
CreateDec ... 6633
CreateThumb ... 6634
OnClickInc ... 6635
OnClickDec ... 6636
OnShow ... 6637
OnHide ... 6638
OnChangePos ... 6639
OnThumbDragStart ... 6640
OnThumbDragProcess ... 6641
OnThumbDragEnd ... 6642
CalcPos ... 6643
CScrollV ... 6644
CreateInc ... 6650
CreateDec ... 6651
CreateThumb ... 6652
OnResize ... 6653
OnChangePos ... 6654
OnThumbDragStart ... 6655
OnThumbDragProcess ... 6656
OnThumbDragEnd ... 6657
CalcPos ... 6658
CScrollH ... 6659
CreateInc ... 6665
CreateDec ... 6666
CreateThumb ... 6667
OnResize ... 6668
OnChangePos ... 6669
OnThumbDragStart ... 6670
OnThumbDragProcess ... 6671
OnThumbDragEnd ... 6672
CalcPos ... 6673
CWndClient ... 6674
Create ... 6677
OnEvent ... 6678
ColorBackground ... 6679
ColorBorder ... 6680
BorderType ... 6681
VScrolled ... 6682
HScrolled ... 6683
CreateBack ... 6684
CreateScrollV ... 6685
CreateScrollH ... 6686
OnResize ... 6687

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow ... 6688
OnVScrollHide ... 6689
OnHScrollShow ... 6690
OnHScrollHide ... 6691
OnScrollLineDown ... 6692
OnScrollLineUp ... 6693
OnScrollLineLeft ... 6694
OnScrollLineRight ... 6695
Rebound ... 6696
CListView ... 6697
Create ... 6703
OnEvent ... 6704
TotalView ... 6705
AddItem ... 6706
Select ... 6707
SelectByText ... 6708
SelectByValue ... 6709
Value ... 6710
CreateRow ... 6711
OnResize ... 6712
OnVScrollShow ... 6713
OnVScrollHide ... 6714
OnScrollLineDown ... 6715
OnScrollLineUp ... 6716
OnItemClick ... 6717
Redraw ... 6718
RowState ... 6719
CheckView ... 6720
CComboBox ... 6721
Create ... 6727
OnEvent ... 6728
AddItem ... 6729
ListViewItems ... 6730
Select ... 6731
SelectByText ... 6732
SelectByValue ... 6733
Value ... 6734
CreateEdit ... 6735
CreateButton ... 6736
CreateList ... 6737
OnClickEdit ... 6738
OnClickButton ... 6739
OnChangeList ... 6740
ListShow ... 6741
ListHide ... 6742
CCheckBox ... 6743
Create ... 6750
OnEvent ... 6751
Text ... 6752
Color ... 6753
Checked ... 6754
Value ... 6755
CreateButton ... 6756
CreateLabel ... 6757
OnClickButton ... 6758
OnClickLabel ... 6759
CCheckGroup ... 6760
Create ... 6766

© 2000-2025, MetaQuotes Ltd.

OnEvent ... 6767
AddItem ... 6768
Value ... 6769
CreateButton ... 6770
OnVScrollShow ... 6771
OnVScrollHide ... 6772
OnScrollLineDown ... 6773
OnScrollLineUp ... 6774
OnChangeItem ... 6775
Redraw ... 6776
RowState ... 6777
CRadioButton ... 6778
Create ... 6780
OnEvent ... 6781
Text ... 6782
Color ... 6783
State ... 6784
CreateButton ... 6785
CreateLabel ... 6786
OnClickButton ... 6787
OnClickLabel ... 6788
CRadioGroup ... 6789
Create ... 6795
OnEvent ... 6796
AddItem ... 6797
Value ... 6798
CreateButton ... 6799
OnVScrollShow ... 6800
OnVScrollHide ... 6801
OnScrollLineDown ... 6802
OnScrollLineUp ... 6803
OnChangeItem ... 6804
Redraw ... 6805
RowState ... 6806
Select ... 6807
CSpinEdit ... 6808
Create ... 6814
OnEvent ... 6815
MinValue ... 6816
MaxValue ... 6817
Value ... 6818
CreateEdit ... 6819
CreateInc ... 6820
CreateDec ... 6821
OnClickInc ... 6822
OnClickDec ... 6823
OnChangeValue ... 6824
CDialog ... 6825
Create ... 6828
OnEvent ... 6829
Caption ... 6830
Add ... 6831
CreateWhiteBorder ... 6832
CreateBackground ... 6833
CreateCaption ... 6834
CreateButtonClose ... 6835
CreateClientArea ... 6836
OnClickCaption ... 6837

© 2000-2025, MetaQuotes Ltd.

OnClickButtonClose ... 6838
ClientAreaVisible ... 6839
ClientAreaLeft ... 6840
ClientAreaTop ... 6841
ClientAreaRight ... 6842
ClientAreaBottom ... 6843
ClientAreaWidth ... 6844
ClientAreaHeight ... 6845
OnDialogDragStart ... 6846
OnDialogDragProcess ... 6847
OnDialogDragEnd ... 6848
CAppDialog ... 6849
Create ... 6852
Destroy ... 6853
OnEvent ... 6854
Run ... 6855
ChartEvent ... 6856
Minimized ... 6857
IniFileSave ... 6858
IniFileLoad ... 6859
IniFileName ... 6860
IniFileExt ... 6861
CreateCommon ... 6862
CreateExpert ... 6863
CreateIndicator ... 6864
CreateButtonMinMax ... 6865
OnClickButtonClose ... 6866
OnClickButtonMinMax ... 6867
OnAnotherApplicationClose ... 6868
Rebound ... 6869
Minimize ... 6870
Maximize ... 6871
CreateInstanceId ... 6872
ProgramName ... 6873
SubwinOff ... 6874
36 Переход с MQL4 ... 6875
37 Список функций языка MQL5 ... 6879
38 Список констант языка MQL5 ... 6918

© 2000-2025, MetaQuotes Ltd.

Справочник MQL5

MetaQuotes Language 5 (MQL5) — язык программирования технических индикаторов, торговых роботов и вспомогательных приложений для автоматизации торговли на финансовых рынках. MQL5 является современным языком высокого уровня и разработан компанией MetaQuotes для собственной торгово-информационной платформы. Синтаксис языка максимально близок к C++ и позволяет писать программы в стиле объектно-ориентированного программирования (ООП).

Для написания программ на MQL5 в составе торговой платформы предоставляется среда разработки MetaEditor со всеми современными инструментами для написания кода, включающими в себя шаблоны, сниппеты, отладку, профилировку, автозавершение и встроенное версионное хранилище MQL5 Storage.

Поддержка и развитие языка осуществляется на сайте MQL5 Algotrading community, где находится обширная библиотека бесплатных кодов и множество статей. Эти статьи охватывают все темы современного трейдинга: нейронные сети, статистика и анализ, высокочастотная торговля, арбитраж, тестирование и оптимизация торговых стратегий, использование роботов для автоматизации торговли и многое другое.

Трейдеры и разработчики MQL5-программ могут общаться на форуме, проводить заказы во Фрилансе, покупать и продавать защищенные программы в Маркете - магазине готовых приложений для автотрейдинга.

Язык MQL5 содержит специализированные торговые функции и предопределенные обработчики событий для написания советников (Expert Advisors). Советники автоматически управляют торговыми процессами на основе заложенных в них торговых правил. Также на MQL5 можно создавать собственные технические индикаторы (Custom Indicators), скрипты (Scripts) и библиотеки функций (Libraries).

Справочник MQL5 содержит разбитые на категории функции, операции, зарезервированные слова, другие конструкции языка и позволяет узнать описание каждого используемого элемента, входящего в состав языка. Также в справочнике приведено описание классов из состава Стандартной библиотеки для создания торговых стратегий, панелей управления, пользовательской графики и работы с файлами.

Отдельно от справочника в CodeBase опубликована библиотека численного анализа ALGLIB, которая позволяет решать множество математических задач.

Учебники по алготрейдингу

Начинать изучение нового всегда непросто. В помощь новичкам мы выпустили два учебника по программированию на языке MQL5, предназначенных для всех, кто хочет освоить создание торговых роботов и приложений для алгоритмической торговли.

Наши учебники предлагают читателям систематическое и структурированное изложение материала, что существенно облегчает процесс изучения MQL5. Подробные примеры кода, шаг за шагом разъясняющие создание торговых роботов и приложений, позволяют глубже понять и освоить тонкости алготрейдинга. Книги содержат множество практических упражнений, которые помогут закрепить полученные знания и развить навыки программирования в реальных торговых условиях.

© 2000-2025, MetaQuotes Ltd.

"Программирование на MQL5 для трейдеров" — самый полный и детальный учебник по MQL5, подходит программистам любого уровня. Новички познакомятся с основами: изучат инструменты разработки и базовые понятия в программировании. Создадут первую программу, скомпилируют и запустят ее в торговой платформе MetaTrader 5. Пользователи с опытом программирования на других языках могут сразу же перейти к прикладной части: возможностям MQL5 для создания торговых роботов и аналитических приложений.

"Нейросети в алготрейдинге на MQL5" — учебное пособие по использованию методов машинного обучения при создании торговых роботов в платформе MetaTrader 5. Вы последовательно познакомитесь с основами нейросетей и возможностями их применения в алготрейдинге. По мере продвижения вы создадите и обучите собственный искусственный интеллект, постепенно дополняя его новыми функциями. Помимо изучения MQL5, вы научитесь писать программы на Python и OpenCL, а также узнаете о встроенных методах матриц и векторов, которые позволяют решать сложные математические задачи с помощью лаконичного и быстрого кода.

Статьи по разработке торговых приложений

Отличным пособием по изучению всех возможностей языка являются статьи по MQL5, в которых рассматривается множество практических задач по алготрейдингу. Для удобства поиска все статьи разделены по категориям: Примеры, Эксперты, Машинное обучение и так далее. Ежемесячно на сайте MQL5 Algotrading community выходят десятки новых статей, которые пишут сами трейдеры для других трейдеров. Читайте и обсуждайте статьи, в них обсуждаются все аспекты современного алготрейдинга. Для новичков мы составили список из 16 рекомендуемых статьей для быстрого погружения в MQL5.

Виды приложений в MQL5

Для выполнения конкретных задач по автоматизации торговых операций MQL5-программы разделены на пять специализированных типов:

- **Советник** — автоматическая торговая система, имеющая привязку к определенному графику. Советник содержит в себе функции-обработчики предопределенных событий, при наступлении которых выполняются соответствующие элементы торговой стратегии. Примеры таких событий - инициализация и деинициализация программы, приход нового тика, срабатывание таймера, изменение в стакане цен, события графика и пользовательские события.
  Советник может не только вычислять торговые сигналы по заложенным правилам, но и автоматически совершать сделки на торговом счете, направляя их прямо на торговый сервер. Советники хранятся в директории *<каталог_терминала>\MQL5\Experts*.
- **Пользовательский индикатор** — это технический индикатор, написанный пользователем в дополнение к индикаторам, уже интегрированным в торговую платформу. Пользовательские индикаторы, также как и встроенные, не могут автоматически торговать и предназначены только для реализации аналитических функций. Пользовательские индикаторы могут использовать в своих расчетах значения других индикаторв, а также сами могут вызываться в советниках.
  Пользовательские индикаторы хранятся в директории *<каталог_терминала>\MQL5\Indicators*.
- **Скрипт** — программа, предназначенная для одноразового выполнения каких-либо действий. В отличие от экспертов, скрипты не обрабатывают никаких событий, кроме события запуска. Для

© 2000-2025, MetaQuotes Ltd.

работы скрипта в его коде обязательно должна быть функция-обработчик OnStart. Скрипты хранятся в директории *<каталог_терминала>\MQL5\Scripts*.

- **Сервис** — программа, которая в отличие от индикаторов, советников и скриптов для своей работы не требует привязки к графику. Как и скрипты, сервисы не обрабатывают никаких событий, кроме события запуска. Для запуска сервиса в его коде обязательно должна быть функция-обработчик OnStart. Сервисы не принимают никаких других событий кроме Start, но могут сами отправлять графикам пользовательские события с помощью EventChartCustom. Сервисы хранятся в директории *<каталог_терминала>\MQL5\Services*.
- **Библиотека** — библиотека пользовательских функций, предназначенная для хранения и распространения часто используемых блоков пользовательских программ. Библиотеки не могут самостоятельно запускаться на выполнение.
  Библиотеки хранятся в директории *<каталог_терминала>\MQL5\Libraries*
- **Включаемый файл** — исходный текст часто используемых блоков пользовательских программ. Такие файлы могут включаться в исходные тексты экспертов, скриптов, пользовательских индикаторов и библиотек на этапе компиляции. Использование включаемых файлов более предпочтительно, чем использование библиотек, из-за дополнительных накладных расходов при вызове библиотечных функций.
  Включаемые файлы могут находиться в той же директории, что и исходный файл, в этом случае используется директива #include с двойными кавычками. Другое место хранения включаемых файлов - в директории *<каталог_терминала>\MQL5\Include*, в этом случае используется директива *#include* с угловыми скобками.

© 2000-2025, MetaQuotes Ltd

Основы языка

Язык MetaQuotes Language 5 (MQL5) является объектно-ориентированным языком программирования высокого уровня и предназначен для написания автоматических торговых стратегий, пользовательских технических индикаторов для анализа разнообразных финансовых рынков. Он позволяет не только писать разнообразные экспертные системы, предназначенные для работы в режиме реального времени, но и создавать собственные графические инструменты, помогающие принимать торговые решения.

MQL5 основан на концепции широко распространенного языка программирования C++, по сравнению с MQL4 в нем добавлены перечисления, структуры, классы и обработка событий. Благодаря расширению числа встроенных основных типов, взаимодействие исполняемых программ на MQL5 с другими приложениями посредством dll максимально облегчено. Синтаксис языка MQL5 подобен синтаксису C++, и это позволяет легко переносить на него программы из современных языков программирования.

Для целей изучения языка MQL5 все темы сгруппированы по следующим разделам:

- Синтаксис
- Типы данных
- Операции и выражения
- Операторы
- Функции
- Переменные
- Препроцессор
- Объектно-ориентированное программирование
- Пространства имен

© 2000-2025, MetaQuotes Ltd.

Синтаксис

Синтаксически язык программирования торговых стратегий MQL5 очень похож на язык программирования C++, за исключением некоторых возможностей:

- отсутствует адресная арифметика;
- отсутствует оператор goto;
- нельзя объявить анонимное перечисление;
- нет множественного наследования.

Смотри также

Перечисления, Структуры и классы, Наследование

© 2000-2025, MetaQuotes Ltd.

Комментарии

Многострочные комментарии начинаются парой символов /* и заканчиваются парой */. Данные комментарии не могут быть вложенными. Однострочные комментарии начинаются парой символов //, заканчиваются символом новой строки и могут быть вложены в многострочные комментарии. Комментарии разрешены везде, где возможны пробелы, и допускают любое число пробелов.

Примеры:

```
//--- Однострочный комментарий
/*  Многостроч-
    ный          // Вложенный однострочный комментарий
    комментарий
*/
```

© 2000-2025, MetaQuotes Ltd.

Идентификаторы

Идентификаторы используются в качестве имен для переменных и функций. Длина идентификатора не может превышать 63 знака.

Допустимые символы при написании идентификатора: цифры 0-9, латинские прописные и строчные буквы a-z и A-Z, распознаваемые как разные символы, символ подчеркивания (_). Первый символ не может быть цифрой.

Идентификатор не должен совпадать с зарезервированным словом.

Примеры:

```
NAME1 name1 Total_5 Paper
```

Смотри также

Переменные, Функции

© 2000-2025, MetaQuotes Ltd.

Зарезервированные слова

Перечисленные ниже идентификаторы фиксируются как зарезервированные слова, каждому из которых соответствует определенное действие, и в другом смысле не могут использоваться:

Типы данных

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

Спецификаторы доступа

const	private	virtual
delete	protected	
override	public	

Классы памяти

extern	input	static

Операторы

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

Прочие

this	#define	#import
true	#ifdef	#include
false	#ifndef	#property

© 2000-2025, MetaQuotes Ltd.

this	#define	#import
template	#else	group
typename	#endif	namespace

© 2000-2025, MetaQuotes Ltd.

Типы данных

Любая программа оперирует данными. Данные могут быть различных типов в зависимости от назначения. Например, для доступа к элементам массива используются данные целочисленного типа. Ценовые данные имеют тип двойной точности с плавающей точкой. Это связано с тем, что в языке MQL5 не предусмотрено специального типа для ценовых данных.

Данные разного типа обрабатываются с разной скоростью. Целочисленные данные обрабатываются быстрее всего. Для обработки данных двойной точности используется специальный сопроцессор. Однако из-за сложности внутреннего представления данных с плавающей точкой, они обрабатывается дольше, чем целочисленные.

Дольше всего обрабатываются строковые данные. Это связано с динамическим распределением-перераспределением оперативной памяти компьютера.

Основные типы данных:

- целые (char, short, int, long, uchar, ushort, uint, ulong)
- логические (bool)
- литералы (ushort)
- строки (string)
- с плавающей точкой (double, float)
- цвет (color)
- дата и время (datetime)
- перечисления (enum)

Сложные типы данных:

- структуры;
- классы.

В терминах ООП сложные типы данных называются абстрактными типами данных.

Типы color и datetime имеют смысл только для удобства представления и ввода параметров, задаваемых извне - из таблицы свойств советника или пользовательского индикатора (вкладка "Inputs"). Данные типов color и datetime представляются в виде целых чисел. Целые типы вместе с типами с плавающей точкой называются арифметическими (числовыми) типами.

В выражениях используется неявное приведение типов, если не указано явное приведение.

Смотри также

Приведение типов

© 2000-2025, MetaQuotes Ltd.

Целые типы

Целые типы представлены в языке MQL5 одиннадцатью видами. Некоторые из типов могут использоваться вместе с другими, если этого требует логика программы, но при этом необходимо иметь ввиду правила преобразования типов.

В таблице приведены характеристики каждого типа. Кроме того, в последнем столбце для каждого типа указан соответствующий тип в языке программирования C++.

Тип	Размер в байтах	Минимальное значение	Максимальное значение	Аналог в языке C++
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

Значения целых типов можно также представлять в виде числовых констант, цветовых литералов, литералов даты-времени, символьных констант и перечислений.

Дополнительные псевдонимы целочисленных типов (для совместимости с C/C++)

Для упрощения портирования кода из других языков, таких как C и C++, в язык MQL5 были добавлены псевдонимы (алиасы) для стандартных целочисленных типов. Эти алиасы не вводят новые типы, а являются альтернативными именами уже существующих типов в MQL5. Их можно использовать во всех контекстах, где применимы базовые типы.

© 2000-2025, MetaQuotes Ltd.

Тип	Соответствует типу MQL5	Размер (бит)	Знак	Минимальное значение	Максимальное значение
int8_t	char	8	знаковый	-128	127
uint8_t	uchar	8	беззнаковый	0	255
int16_t	short	16	знаковый	-32 768	32 767
uint16_t	ushort	16	беззнаковый	0	65 535
int32_t	int	32	знаковый	- 2 147 483 648	2 147 483 647
uint32_t	uint	32	беззнаковый	0	4 294 967 295
int64_t	long	64	знаковый	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
uint64_t	ulong	64	беззнаковый	0	18 446 744 073 709 551 615

Использование этих псевдонимов полезно при переносе существующих библиотек и алгоритмов из C/C++ в MQL5, особенно в проектах с высоким уровнем платформенной совместимости. Все новые имена доступны как в скриптах, так и в советниках и индикаторах.

Смотри также

Преобразование данных, Константы числовых типов

© 2000-2025, MetaQuotes Ltd.

Типы char, short, int и long

char

Целый тип char занимает в памяти 1 байт (8 бит) и позволяет выразить в двоичной системе счисления 2^8 значений=256. Тип char может содержать как положительные, так и отрицательные значения. Диапазон изменения значений составляет от -128 до 127.

uchar

Целый тип uchar также занимает в памяти 1 байт, как и тип char, но в отличие от него, uchar предназначен только для положительных значений. Минимальное значение равно нулю, максимальное значение равно 255. Первая буква u в названии типа uchar является сокращением слова unsigned (беззнаковый).

short

Целый тип short имеет размер 2 байта(16 бит) и, соответственно, позволяет выразить множество значений равное 2 в степени 16: 2^16=65 536. Так как тип short является знаковым и содержит как положительные, так и отрицательные значения, то диапазон значений находится между -32 768 и 32 767.

ushort

Беззнаковым типом short является тип ushort, который также имеет размер 2 байта. Минимальное значение равно 0, максимальное значение 65 535.

int

Целый тип int имеет размер 4 байта (32 бита). Минимальное значение -2 147 483 648, максимальное значение 2 147 483 647.

uint

Беззнаковый целый тип uint занимает в памяти 4 байта и позволяет выражать целочисленные значения от 0 до 4 294 967 295.

long

Целый тип long имеет размер 8 байт (64 бита). Минимальное значение -9 223 372 036 854 775 808, максимальное значение 9 223 372 036 854 775 807.

ulong

Целый тип ulong также занимает 8 байт и позволяет хранить значения от 0 до 18 446 744 073 709 551 615.

© 2000-2025, MetaQuotes Ltd.

Примеры:

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

Так как беззнаковые целые типы не предназначены для хранения отрицательных значений, то попытка установить отрицательное значение может привести к неожиданным последствиям. Вот такой невинный скрипт приведет к бесконечному циклу:

```
//--- бесконечный цикл
void OnStart()
  {
   uchar  u_ch;

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

Правильно будет так:

```
//--- правильный вариант
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;
     }
  }
```

Результат:

```
   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
```

© 2000-2025, MetaQuotes Ltd.

```
   ch= -115  u_ch= 141
   ch= -114  u_ch= 142
   ch= -113  u_ch= 143
   ch= -112  u_ch= 144
   ch= -111  u_ch= 145
    ...
```

Примеры:

```
//--- отрицательные значения нельзя хранить в беззнаковых типах
uchar  u_ch=-120;
ushort u_sh=-5000;
uint   u_in=-401280;
```

Шестнадцатеричные: цифры 0-9, буквы a-f или A-F для значений 10-15; начинаются с 0x или 0X.

Примеры:

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

Для целочисленных переменных значения можно задавать в бинарном виде с помощью префикса B. Например, можно закодировать рабочие часы торговой сессии в переменную типа int и использовать информацию о них согласно требуемому алгоритму:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- для рабочих часов ставим 1, для нерабочих указываем 0
   int AsianSession   =B'111111111'; // азиатская сессия с 0:00 часов до 9:00
   int EuropeanSession=B'111111111000000000'; // европейская сессия  9:00 - 18:00
   int AmericanSession =B'111111110000000000000011'; // американская 16:00 - 02:00
//--- выведем числовые значения сессий
   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);
//--- а теперь выведем строковые представления рабочих часов сессий
   Print("Asian session ",GetHoursForSession(AsianSession));
   Print("European session ",GetHoursForSession(EuropeanSession));
   Print("American session ",GetHoursForSession(AmericanSession));
//---
  }
//+------------------------------------------------------------------+
//| возвращает рабочие часы сессии в строковом виде                  |
//+------------------------------------------------------------------+
string GetHoursForSession(int session)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- для проверки используем битовые операции AND и сдвиг на один бит влево <<=1
//--- начинаем проверять с самого младшего бита
   int bit=1;
   string out="working hours: ";
//--- будем проверять все 24 бита, начиная с нулевого до 23 включительно
   for(int i=0;i<24;i++)
     {
      //--- получим состояние бита bit в числе number
      bool workinghour=(session&bit)==bit;
      //--- добавим номер часа в сообщение
      if(workinghour )out=out+StringFormat("%d ",i);
      //--- сдвигаем на один бит влево для проверки значения следующего
      bit<<=1;
     }
//--- сформированная строка
   return out;
  }
```

Смотри также

Приведение типов

© 2000-2025, MetaQuotes Ltd.

Символьные константы

Символы, как элемент строки, в MQL5 - это индексы в наборе символов Unicode. Они являются 16-разрядными значениями, которые можно преобразовывать в целые числа и с которыми можно манипулировать целочисленными операциями, такими как сложение и вычитание.

Любой одиночный символ, заключенный в одинарные кавычки, или шестнадцатеричный ASCII-код символа в виде '\x10' является символьной константой и имеет тип ushort. Например, запись вида '0' представляет из себя числовое значение 30, соответствующее индексу, по которому в таблице символов располагается символ ноль.

Пример:

```
void OnStart()
  {
//--- определим символьные константы
   int symbol_0='0';
   int symbol_9=symbol_0+9; // получим символ '9'
//--- выведем значения констант
   printf("В десятичном виде: symbol_0 = %d,  symbol_9 = %d",symbol_0,symbol_9);
   printf("В шестнадцетеричном виде: symbol_0 = 0x%x,  symbol_9 = 0x%x",symbol_0,symbo
//--- заведем константы в строку
   string test="";
   StringSetCharacter(test,0,symbol_0);
   StringSetCharacter(test,1,symbol_9);
//--- а вот как они выглядят в строке
   Print(test);
  }
```

Обратная косая черта является управляющим символом для компилятора при разборе константных строк и символьных констант в исходном тексте программы. Некоторые символы, например, одинарные кавычки ('), двойные кавычки ("), обратная косая черта (\) и управляющие символы можно представлять комбинацией символов, начинающейся с обратной косой черты(\), в соответствии с приводимой ниже таблицей:

Название символа	Мнемокод или изображение	Запись в MQL5	Числовое значение
новая строка (перевод строки)	LF	'\n'	10
горизонтальная табуляция	HT	'\t'	9
возврат каретки	CR	'\r'	13
обратная косая черта	\	'\\'	92
одинарная кавычка	'	'\''	39
двойная кавычка	"	'\"'	34

© 2000-2025, MetaQuotes Ltd.

Название символа	Мнемокод или изображение	Запись в MQL5	Числовое значение
шестнадцатеричный код	hhhh	'\xhhhh'	от 1до 4 шестнадцатеричных знаков
десятичный код	d	'\d'	десятичное число от 0 до 65535

Если за обратной косой чертой следует символ, отличный от перечисленных, результат не определен.

Пример

```
void OnStart()
  {
//--- объявим символьные константы
   int a='A';
   int b='$';
   int c='©';       // код 0xA9
   int d='\xAE';    // код символа ®
//--- выведем константы на печать
   Print(a,b,c,d);
//--- добавим символ в строку
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- заменим символ в строке
   StringSetCharacter(test,0,b);
   Print(test);
//--- заменим символ в строке
   StringSetCharacter(test,0,c);
   Print(test);
//--- заменим символ в строке
   StringSetCharacter(test,0,d);
   Print(test);
//--- представим символы в виде числа
   int a1=65;
   int b1=36;
   int c1=169;
   int d1=174;
//--- добавим символ в строку
   StringSetCharacter(test,1,a1);
   Print(test);
//--- заменим символ в строке
   StringSetCharacter(test,1,b1);
   Print(test);
//--- заменим символ в строке
   StringSetCharacter(test,1,c1);
```

© 2000-2025, MetaQuotes Ltd.

```
   Print(test);
//--- заменим символ в строке
   StringSetCharacter(test,1,d1);
   Print(test);
  }
```

Как уже говорилось выше, значение символьной константы (или переменной) представляет собой индекс в таблице символов, а так как индекс является целым числом, то допустимо его записывать разными способами.

```
void OnStart()
  {
//---
   int a=0xAE;      // код символа ® соответствует литералу '\xAE'
   int b=0x24;      // код символа $ соответствует литералу '\x24'
   int c=0xA9;      // код символа © соответствует литералу '\xA9'
   int d=0x263A;    // код символа ☺ соответствует литералу '\x263A'
//--- выведем значения
   Print(a,b,c,d);
//--- добавим символ в строку
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- заменим символ в строке
   StringSetCharacter(test,0,b);
   Print(test);
//--- заменим символ в строке
   StringSetCharacter(test,0,c);
   Print(test);
//--- заменим символ в строке
   StringSetCharacter(test,0,d);
   Print(test);
//--- коды мастей
   int a1=0x2660;
   int b1=0x2661;
   int c1=0x2662;
   int d1=0x2663;
//--- добавим символ пикей
   StringSetCharacter(test,1,a1);
   Print(test);
//--- добавим символ червей
   StringSetCharacter(test,2,b1);
   Print(test);
//--- добавим символ бубей
   StringSetCharacter(test,3,c1);
   Print(test);
//--- добавим символ треф
   StringSetCharacter(test,4,d1);
   Print(test);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- Пример символьных литералов в строке
   test="Дама\x2660Туз\x2662";
   printf("%s",test);
  }
```

Внутреннее представление символьного литерала - тип ushort. Символьные константы могут принимать значения от 0 до 65535.

Смотри также

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

© 2000-2025, MetaQuotes Ltd.

Тип datetime

Тип datetime предназначен для хранения даты и времени в виде количества секунд, прошедших с 01 января 1970 года. Занимает в памяти 8 байт.

Константы даты и времени могут быть представлены в виде литеральной строки, которая состоит из 6 частей, представляющих числовое значение года, месяца, числа (либо числа, месяца, года), часа, минуты и секунды. Константа обрамляется одинарными кавычками и начинается с символа D.

Диапазон значений от 1 января 1970 года до 31 декабря 3000 года. Может опускаться либо дата (год, месяц, число), либо время (часы, минуты, секунды), либо все вместе.

При литеральном задании даты желательно указывать год, месяц и день, иначе компилятор выдаст предупреждения о неполной литеральной записи.

Примеры:

```
datetime NY=D'2015.01.01 00:00';      // время наступления 2015 года
datetime d1=D'1980.07.19 12:30:27';   // год месяц день часы минуты секунды
datetime d2=D'19.07.1980 12:30:27';   // равнозначно D'1980.07.19 12:30:27';
datetime d3=D'19.07.1980 12';         // равнозначно D'1980.07.19 12:00:00'
datetime d4=D'01.01.2004';            // равнозначно D'01.01.2004 00:00:00'
datetime compilation_date=__DATE__;              // дата компиляции
datetime compilation_date_time=__DATETIME__;     // дата и время компиляции
datetime compilation_time=__DATETIME__-__DATE__;// время компиляции
//--- примеры объявлений, на которые будут получены предупреждения компилятора
datetime warning1=D'12:30:27';        // равнозначно D'[дата компиляции] 12:30:27'
datetime warning2=D'';                // равнозначно __DATETIME__
```

Смотри также

Структура даты, Дата и время, TimeToString, StringToTime

© 2000-2025, MetaQuotes Ltd.

Тип color

Тип color предназначен для хранения информации о цвете и занимает в памяти 4 байта. Первый байт не учитывается, остальные 3 байта содержат RGB-составляющие.

Цветовые константы могут быть представлены тремя различными способами: литерально, целочисленно или при помощи имени (только для именованных Web-цветов).

Литеральное представление состоит из трех частей, представляющих числовые значения интенсивности трех основных компонент цвета: красной (red), зеленой (green), синей (blue). Константа начинается с символа C и обрамляется одинарными кавычками. Числовые значения интенсивности компоненты цвета лежат в диапазоне от 0 до 255.

Целочисленное представление записывается в виде шестнадцатеричного или десятичного числа. Шестнадцатеричное число имеет вид 0x00BBGGRR, где RR - значение интенсивности красной компоненты цвета, GG - зеленной, а BB - синей. Десятичные константы не имеют прямого отражения в RGB. Они представляют собой десятичное значение шестнадцатеричного целочисленного представления.

Именованные цвета отражают так называемый набор Web-цветов.

Примеры:

```
//--- литералы
C'128,128,128'    // серый
C'0x00,0x00,0xFF' // синий
//названия цветов
clrRed            // красный
clrYellow         // желтый
clrBlack          // черный
//--- целочисленные представления
0xFFFFFF          // белый
16777215          // белый
0x008000          // зеленый
32768             // зеленый
```

Смотри также

Набор Web-цветов, ColorToString, StringToColor, Приведение типов

© 2000-2025, MetaQuotes Ltd.

Тип bool

Тип bool предназначен для хранения логических значений true (истина) или false (ложь), числовое представление которых 1 или 0 соответственно.

Примеры:

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

Внутреннее представление - целое число размером 1 байт. Необходимо отметить, что в логических выражениях допустимо использовать вместо типа bool другие целые или вещественные типы или выражения этих типов, компилятор не выдаст ошибки. В таком случае значение ноль будет интерпретировано как false, а все остальные значения как true.

Примеры:

```
   int i=5;
   double d=-2.5;
   if(i)
      Print("i = ",i," и имеет значение true");
   else
      Print("i = ",i," и имеет значение false");

   if(d)
      Print("d = ",d," и имеет значение true");
   else
      Print("d = ",d," и имеет значение false");

   i=0;
   if(i)
      Print("i = ",i," и имеет значение true");
   else
      Print("i = ",i," и имеет значение false");

   d=0.0;
   if(d)
      Print("d = ",d," и имеет значение true");
   else
      Print("d = ",d," и имеет значение false");

//--- результаты выполнения
//    i= 5 и имеет значение true
//    d= -2.5 и имеет значение true
//    i= 0 и имеет значение false
//    d= 0 и имеет значение false
```

Смотри также

Логические операции, Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Перечисления

Данные перечислимого типа enum относятся к некоторому ограниченному множеству данных. Определение перечислимого типа:

```
enum имя_перечислимого_типа
  {
   список_значений
  };
```

Список значений представляет из себя список идентификаторов именованных констант, разделенных запятыми.

Пример:

```
enum months  // перечисление именованных констант
   {
    January,
    February,
    March,
    April,
    May,
    June,
    July,
    August,
    September,
    October,
    November,
    December
   };
```

После объявления перечисления появляется новый целочисленный 4-байтовый тип данных. Объявление нового типа данных позволяет компилятору строго контролировать типы передаваемых параметров, так как перечисление вводит новые именованные константы. В приведенном примере именованная константа January имеет значение 0, February имеет значение 1, December имеет значение 11.

Правило: если именованной константе - члену перечисления явно не присвоено конкретное значение, то ее значение будет сформировано автоматически. Если это первый член перечисления, то будет присвоено значение 0. Для всех последующих членов значения будет вычисляться на основе значения предыдущего члена путем прибавления единицы.

Пример:

```
enum intervals  // перечисление именованных констант
   {
    month=1,     // интервал в один месяц
    two_months,  // два месяца
    quarter,     // три месяца - квартал
    halfyear=6,  // полугодие
    year=12,     // год - 12 месяцев
   };
```

© 2000-2025, MetaQuotes Ltd.

Примечания

- В отличие от C++, размер внутреннего представления перечислимого типа в MQL5 всегда составляет 4 байта. То есть, sizeof(months) вернет значение 4.
- В отличие от C++, в MQL5 нельзя объявить анонимное перечисление. То есть, после ключевого слова enum всегда должно быть указано уникальное имя.

Смотри также

Приведение типов

© 2000-2025, MetaQuotes Ltd.

Вещественные типы (double, float)

Вещественные типы (или типы с плавающей точкой) представляют значения, имеющие дробную часть. В языке MQL5 есть два типа для чисел с плавающей точкой. Способ представления вещественных чисел в машинной памяти определен стандартом IEEE 754 и не зависит от платформ, операционных систем и языков программирования.

Тип	Размер в байтах	Минимальное положительное значение	Максимальное значение	Аналог в C++
float	4	1.175494351e-38	3.402823466e+38	float
double	8	2.2250738585072014e-308	1.7976931348623158e+308	double

double

Вещественный тип чисел double занимает 64 бита (1 знаковый бит, 11 бит экспоненты и 52 бита мантиссы).

float

Вещественный тип чисел float занимает 32 бита (1 знаковый бит, 8 бит экспоненты и 23 бита мантиссы).

vector

Одномерный массив чисел типа double. Память под данные распределяется динамически. Свойства вектора могут быть получены при помощи методов, при этом размер вектора может быть изменен. В шаблонных функциях можно использовать запись vector<double>.

vectorf

Одномерный массив чисел типа float, может использоваться вместо vector, если потеря точности не имеет значения. В шаблонных функциях можно использовать запись vector<float>.

vectorc

Одномерный массив чисел типа complex, предназначен для работы с комплесными числами. В шаблонных функциях можно использовать запись vector<complex>. На данный момент операции над векторами типа vectorc еще не реализованы.

matrix

© 2000-2025, MetaQuotes Ltd.

Матрица — двумерный массив чисел типа double. Память под элементы матрицы распределяется динамически. Свойства матрицы могут быть получены при помощи методов, при этом размер вектора может быть изменен. В шаблонных функциях можно использовать запись matrix<double>.

matrixf

Двумерный массив чисел типа float, может использоваться вместо matrix, если потеря точности не имеет значения. В шаблонных функциях можно использовать запись matrix<float>.

matrixc

Двумерный массив чисел типа complex, предназначен для работы с комплесными числами. В шаблонных функциях можно использовать запись matrix<complex>. На данный момент операции над матрицами типа matrixc еще не реализованы.

Имя double означает, что точность этих чисел вдвое превышает точность чисел типа float. В большинстве случаев тип double является наиболее удобным. Ограниченной точности чисел float во многих случаях попросту недостаточно. Причина, по которой тип float все еще используется, - экономия памяти при хранении (это важно для больших массивов вещественных чисел).

Константы с плавающей точкой состоят из целой части, точки (.) и дробной части. Целая и дробная части представляют собой последовательности десятичных цифр.

Примеры:

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

Существует научный способ записи вещественных констант, зачастую этот способ записи более компактный, чем традиционный.

Пример:

```
double c1=1.12123515e-25;
double c2=0.00000000000000000000000112123515; // 24 нуля после десятичной точки

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

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

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

Необходимо помнить, что вещественные числа хранятся в памяти компьютера с некоторой ограниченной точностью в двоичной системе счисления, в то время как общепринятой в использовании является десятичная система счисления. Поэтому многие числа, которые точно

© 2000-2025, MetaQuotes Ltd.

записываются в десятичной системе, в двоичной системе можно записать только в виде бесконечной дроби.

Например, числа 0.3 и 0.7 представлены в компьютере бесконечными дробями, в то время как число 0.25 хранится точно, так как представляет из себя степень двойки.

В связи с этим, категорически не рекомендуется сравнивать между собой два вещественных числа на равенство, так как такое сравнение не является корректным.

Пример:

```
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");
// Результат: 1/3 + 4/3 != 5/3
  }
```

Если все же необходимо сравнить на равенство два вещественных числа, то можно сделать это двумя разными способами. Первый способ заключается в сравнении разницы между двумя числами с какой-то малой величиной, задающей точность сравнения.

Пример:

```
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
```

© 2000-2025, MetaQuotes Ltd.

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

Необходимо отметить, что значение параметра epsilon в приведенном примере не может быть меньше предопределенной константы DBL_EPSILON. Значение этой константы 2.2204460492503131e-016. Для типа float соответствующая константа FLT_EPSILON = 1.192092896e-07. Смысл этих значений таков, что это наименьшее значение, удовлетворяющее условию 1.0+DBL_EPSILON != 1.0 (для чисел типа float 1.0+FLT_EPSILON != 1.0).

Второй способ предполагает сравнивать нормализованную разность двух вещественных чисел с нулевым значением. Сравнивать разность нормализованных чисел с нулём бесполезно, так как в результате любой математической операции с нормализованными числами результат получается ненормализованным.

Пример:

```
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,"equals",f_val);
   else
      Print("Different: d_val = ",DoubleToString(d_val,16),"  f_val = ",DoubleToStrinc
// Результат: Different: d_val= 0.3000000000000000   f_val= 0.3000000119209290
  }
```

В результате некоторых операций математического сопроцессора может получиться недействительное вещественное число, которое нельзя использовать в математических операциях и операциях сравнения, так как результат выполнения операций над недействительными вещественными числами неопределен. Например, при попытке вычислить арксинус от 2, результатом будет минус бесконечность.

Пример:

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

Кроме минус бесконечности существуют плюс бесконечность и NaN (не число). Чтобы определить, что данное число недействительно, можно использовать функцию MathIsValidNumber(). По стандарту IEEE они имеют специальное машинное представление. Например, плюс бесконечность для типа double имеет битовое представление 0x7FF0 0000 0000 0000.

© 2000-2025, MetaQuotes Ltd.

Примеры:

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

//--- начнем
   str1 s1;
   str2 s2;
//---
   s1.d=MathArcsin(2.0);         // получим недействительное число -1.#IND
   s2=s1;
   printf("1.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFFF000000000000;      // недействительное число -1.#QNAN
   s1=s2;
   printf("2.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF7000000000000;      // наиобльшее нечисло SNaN
   s1=s2;
   printf("3.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF8000000000000;      // наименьшее нечисло QNaN
   s1=s2;
   printf("4.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FFF000000000000;      // наибольшее нечисло QNaN
   s1=s2;
   printf("5.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF0000000000000;      // плюс бесконечность 1.#INF и наименьшее нечисло SNaN
   s1=s2;
   printf("6.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFF0000000000000;      // минус бесконечность -1.#INF
   s1=s2;
   printf("7.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x8000000000000000;      // отрицательный ноль -0.0
   s1=s2;
   printf("8.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FE0000000000000;      // 0.5
   s1=s2;
```

© 2000-2025, MetaQuotes Ltd.

```
   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;      // наибольшее нормализованное число (MAX_DBL)
   s1=s2;
   printf("11.  %.16e %I64X",s1.d,s2.l);
//---
   s2.l=0x0010000000000000;      // наименьшее положительное нормализованное (MIN_DBL)
   s1=s2;
   printf("12.  %.16e %.16I64X",s1.d,s2.l);
//---
   s1.d=0.7;                     // покажем, что число 0.7 - бесконечная дробь
   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
*/
```

Смотри также

DoubleToString, NormalizeDouble, Константы числовых типов

© 2000-2025, MetaQuotes Ltd.

Комплексное число (complex)

Встроенный тип complex представляет из себя структуру с двумя полями типа double:

```
struct complex
  {
   double            real;    // вещественная часть
   double            imag;    // мнимая часть
  };
```

Тип complex может передаваться по значению в качестве параметра для MQL5-функций (в отличие от обычных структур, которые передаются только по ссылке). Для функций, импортируемых из DLL, тип complex должен передаваться только по ссылке.

Для описания комплексных констант используется суффикс 'i':

```
complex square(complex c)
  {
   return(c*c);
  }
void OnStart()
  {
   Print(square(1+2i));  // в качестве параметра передается константа
  }
// будет выведено "(-3,4)" - это строковое представление комплексного числа
```

Для комплексных чисел на данный момент доступны только простые операции: =, +, -, *, /, +=, -=, *=, /=, ==, !=.

В дальнейшем будут добавлены дополнительные математические функции: получение абсолютного значения, синуса, косинуса и многие другие.

vectorc

Одномерный массив чисел типа complex, предназначен для работы с комплесными числами. В шаблонных функциях можно использовать запись vector<complex>. На данный момент операции над векторами типа vectorc еще не реализованы.

matrixc

Двумерный массив чисел типа complex, предназначен для работы с комплесными числами. В шаблонных функциях можно использовать запись matrix<complex>. На данный момент операции над матрицами типа matrixc еще не реализованы.

© 2000-2025, MetaQuotes Ltd.

Тип string

Тип string предназначен для хранения текстовых строк. Текстовая строка представляет собой последовательность символов в формате Unicode с завершающим нулем на конце. string-переменной может быть назначена строковая константа. Строковая константа представляет собой последовательность символов Unicode, заключенную в двойные кавычки: "Это строковая константа".

Если необходимо ввести в строку двойную кавычку ("), то перед ней надо поставить символ обратной косой черты (\). В строку могут быть введены любые специальные символьные константы, перед которыми стоит символ обратной косой черты (\).

Примеры:

```
string svar="This is a character string";
string svar2=StringSubstr(svar,0,4);
Print("Символ копирайта\t\x00A9");
FileWrite(handle,"эта строка содержит символ перевода строки \n");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

Длинные константные строки для удобства чтения исходного кода можно разбивать на части без операции сложения. Эти части при компиляции автоматически соберутся в одну длинную строку:

```
//--- объявим длинную константную строку
   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>";
//--- выведем в журнал константную строку
   Print(HTML_head);
  }
```

Встроенные методы типа string

Для работы со строками можно использовать строковые функции, функции конвертации и встроенные методы типа string, которые приведены в таблице:

Метод string	Аналог	Описание
Конструктор string(const int len)		Конструирует строку указанной длины
string[] как для чтения, так и для записи. Индекс должен быть в пределах BufferSize()		Обеспечивает доступ к элементу строки по указанному индексу

© 2000-2025, MetaQuotes Ltd.

Метод string	Аналог	Описание
static string string.Init(const int len, const ushort character);	StringInit	Инициализирует строку указанными символами с указанным размером
void string.Fill(const ushort character);	StringFill	Заполняет строку указанным символом
int string.Len();	StringLen	Возвращает число символов в строке
int string.BufferSize();	StringBufferLen	Возвращает размер буфера, распределенного для строки
bool string.SetLen(const int new_len);	StringSetLength	Устанавливает для строки указанную длину в символах
bool string.Reserve(const int buffer_len);	StringReserve	Резервирует в памяти для строки буфер указанного размера
bool string.Add(const string substring);	StringAdd	Присоединяет указанную подстроку с конца
int string.Concatenate(const scalar val1, const scalar val2...);	StringConcatenate	Формирует строку из переданных параметров
array string.Split(const ushort separator, const bool long_separator);	StringSplit	Вовращает массив строк по заданному разделителю
int string.Compare(const string str, const bool case_sensivity);	StringCompare	Сравнивает с указанной строкой и возвращает 1, если первая строка больше второй; 0 - если строки равны; -1 (минус один) - если первая строка меньше второй
string string.Substr(const int start_pos, const int len);	StringSubstr	Извлекает подстроку с указанной позиции
int string.Find(const string substr, const int pos);	StringFind	Возвращает номер позиции, с которой начинается искомая подстрока
void string.ToLower();	StringToLower	Преобразует все символы в строчные (маленькие)
void string.ToUpper();	StringToUpper	Преобразует все символы в прописные (большие)
int string.TrimLeft();	StringTrimLeft	Удаляет символы перевода каретки, пробелы и символы табуляции слева
int string.TrimRight()	StringTrimRight	Удаляет символы перевода каретки, пробелы и символы табуляции справа

© 2000-2025, MetaQuotes Ltd.

Метод string	Аналог	Описание
void string.Double(const double var, const int digits=8);	DoubleToString	Преобразует строку в число типа double
void string.Enum(const enum value);	EnumToString	Преобразование значения перечисления любого типа в строку
void string.Integer(const int value, const int str_len=0, const ushort fill=' ');	IntegerToString	Преобразует строку в число типа long
void string.CharArray(const uchar array[], const int start_pos=0, const int len=-1, const uint cp=CP_ACP);	CharArrayToString	Преобразует часть массива типа uchar в строку
void string.ShortArray(const ushort array[], const int start_pos=0, const int len=-1);	ShortArrayToString	Копирует часть массива типа ushort в строку
void string.Time(const datetime dt,const int mode=TIME_DATE|TIME_MINUTES);	TimeToString	Преобразование значение datetime в строку формата "yyyy.mm.dd hh:mi".
void string.Format(const string format_str);	StringFormat	Форматирует полученные параметры в строку

Смотри также

Преобразование данных, Строковые функции, FileOpen, FileReadString, FileWriteString

© 2000-2025, MetaQuotes Ltd.

Структуры, классы и интерфейсы

Структуры

Структура является набором элементов произвольного типа (кроме типа void). Таким образом, структура объединяет логически связанные данные разных типов.

Объявление структуры

Структурный тип данных определяется следующим описанием:

```
struct имя_структуры
  {
   описания_элементов
  };
```

Имя структуры нельзя использовать в качестве идентификатора (имени переменной или функции). Следует иметь ввиду, что в MQL5 элементы структуры следуют непосредственно друг за другом без выравнивания. В языке C++ такое указание делается компилятору с помощью инструкции

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

Если требуется сделать иное выравнивание в структуре, необходимо использовать вспомогательные члены-"заполнители" нужных размеров.

Пример:

```
struct trade_settings
  {
   uchar  slippage;     // значение допустимого проскальзывания -размер 1 байт
   char   reserved1;    // 1 байт пропуска
   short  reserved2;    // 2 байта пропуска
   int    reserved4;    // еще 4 байта пропуска. Обеспечили выравнивание на границу 8
   double take;         // значения цены фиксации прибыли
   double stop;         // значение цены защитного стопа
  };
```

Такое описание выровненных структур необходимо только для передачи в импортированные dll-функции.

Внимание: данный пример иллюстрирует неправильно спроектированные данные. Лучше было бы сначала объявить данные take и stop большего размера типа double, а затем объявить член slippage типа uchar. В этом случае внутреннее представление данных будет всегда одинаково независимо от значения, указанного в *#pragma pack()*.

Если структура содержит переменные типа string и/или объект динамического массива, то компилятор назначает для такой структуры неявный конструктор, в котором производится обнуление всех членов структуры типа string и правильная инициализация для объекта динамического массива.

Простые структуры

© 2000-2025, MetaQuotes Ltd.

Структуры, которые не содержат строки, объекты класса, указатели и объекты динамических массивов, называются простыми структурами. Переменные простых структур, а также их массивы можно передавать в качестве параметров в импортируемые из DLL функции.

Копирование простых структур допускается только в двух случаях:

- если объекты принадлежат к одному типу структуры
- если объекты связаны между собой линией наследования, то есть одна структура является потомком другой структуры.

Покажем это на примерах, создадим пользовательскую стуктуру CustomMqlTick, идентичную по составу встроенной структуре MqlTick. Попытки скопировать значение объекта MqlTick в объект типа CustomMqlTick компилятор не пропустит. Прямое приведение к нужному типу также вызовет ошибку компиляции:

```
      //--- копировать простые структуры разных типов запрещено
      my_tick1=last_tick;               // компилятор здесь выдаст ошибку

      //--- приводить структуры разного типа друг к другу тоже нельзя
      my_tick1=(CustomMqlTick)last_tick;// компилятор здесь выдаст ошибку
```

Поэтому остается только один вариант - копировать значения членов структуры поэлементно. Но при этом разрешено копировать значения объектов одного и того же типа CustomMqlTick.

```
      CustomMqlTick my_tick1,my_tick2;
      //--- а вот так копировать объекты одной и той же структуры CustomMqlTick можно
      my_tick2=my_tick1;

      //--- создадим массив из объектов простой структуры CustomMqlTick и запишем в не
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
```

В качестве проверки вызывается функция ArrayPrint() для вывода в журнал значения массива *arr[]*.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим такую же структуру, как встроенная MqlTick
   struct CustomMqlTick
     {
      datetime          time;          // Время последнего обновления цен
      double            bid;           // Текущая цена Bid
      double            ask;           // Текущая цена Ask
      double            last;          // Текущая цена последней сделки (Last)
      ulong             volume;        // Объем для текущей цены Last
      long              time_msc;      // Время последнего обновления цен в миллисеку
      uint              flags;         // Флаги тиков
     };
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- получим значения последнего тика
   MqlTick last_tick;
   CustomMqlTick my_tick1,my_tick2;
//--- попытаемся скопировать данные из MqlTick в CustomMqlTick
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      //--- копировать неродственные простые структуры запрещено
      //1. my_tick1=last_tick;               // компилятор здесь выдаст ошибку

      //--- приводить неродственные структуры друг к другу тоже нельзя
      //2. my_tick1=(CustomMqlTick)last_tick;// компилятор здесь выдаст ошибку

      //--- поэтому копируем члены структуры поэлементно
      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;

      //--- а вот так копировать объекты одной и той же структуры CustomMqlTick можно
      my_tick2=my_tick1;

      //--- создадим массив из объектов простой структуры CustomMqlTick и запишем в не
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
      ArrayPrint(arr);
//--- пример вывода значений массива, содержащего объекты типа 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());
  }
```

Второй пример показывает возможности копирования простых структур по линии наследования. Пусть у нас есть базовая стурктура Animal, от которой порождены наследованием структуры Cat и Dog. Мы можем копировать между собой объекты Animal и Cat, Animal и Dog, но не можем копировать между собой Cat и Dog - хотя оба являются потомками структуры Animal.

```
//--- структура для описания собак
struct Dog: Animal
  {
   bool              hunting;       // охотничья порода
  };
```

© 2000-2025, MetaQuotes Ltd.

```
//--- структура для описания кошек
struct Cat: Animal
  {
   bool              home;          // домашняя порода
  };
//--- создадим объекты дочерних структур
   Dog dog;
   Cat cat;
//--- можно копировать от предка к потомку (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // собаки умеют плавать
//--- копировать объекты дочерних структур нельзя (Dog != Cat)
   cat=dog;        // компилятор здесь выдаст ошибку
```

Полный код примера:

```
//--- базовая структура для описания животных
struct Animal
  {
   int               head;          // кол-во голов
   int               legs;          // кол-во ног
   int               wings;         // кол-во крыльев
   bool              tail;          // наличие хвоста
   bool              fly;           // летает
   bool              swim;          // плавает
   bool              run;           // бегает
  };
//--- структура для описания собак
struct Dog: Animal
  {
   bool              hunting;       // охотничья порода
  };
//--- структура для описания кошек
struct Cat: Animal
  {
   bool              home;          // домашняя порода
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим объект базового типа Animal и опишем его
   Animal some_animal;
   some_animal.head=1;
   some_animal.legs=4;
   some_animal.wings=0;
   some_animal.tail=true;
   some_animal.fly=false;
```

© 2000-2025, MetaQuotes Ltd.

```
   some_animal.swim=false;
   some_animal.run=true;
//--- создадим объекты дочерних типов
   Dog dog;
   Cat cat;
//--- можно копировать от предка к потомку (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // собаки умеют плавать
//--- копировать объекты дочерних структур нельзя (Dog != Cat)
   //cat=dog;        // компилятор здесь выдаст ошибку
//--- поэтому можно копировать только поэлементно
   cat.head=dog.head;
   cat.legs=dog.legs;
   cat.wings=dog.wings;
   cat.tail=dog.tail;
   cat.fly=dog.fly;
   cat.swim=false;   // кошки не умеют плавать
//--- копировать значения от потомка к предку можно
   Animal elephant;
   elephant=cat;
   elephant.run=false;// слоны не умеют бегать
   elephant.swim=true;// слоны плавают
//--- создадим массив
   Animal animals[4];
   animals[0]=some_animal;
   animals[1]=dog;
   animals[2]=cat;
   animals[3]=elephant;
//--- выведем на печать
   ArrayPrint(animals);
//--- результат выполнения
/*
        [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
*/
  }
```

Другим способом копировать простые типы является использование объединения, для этого объекты этих структур должны являться членами одного и того же объединения - смотрите пример в union.

Доступ к членам структуры

Имя структуры является новым типом данных и позволяет объявлять переменные этого типа. Структура может быть объявлена только один раз в пределах проекта. Доступ к членам структур производится при помощи операции точка (.).

© 2000-2025, MetaQuotes Ltd.

Пример:

```
struct trade_settings
  {
   double take;          // значения цены фиксации прибыли
   double stop;          // значение цены защитного стопа
   uchar  slippage;      // значение допустимого проскальзывания
  };
//--- создали и проинициализировали переменную типа trade_settings
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

pack для выравнивания полей структур и классов

Специальный атрибут pack позволяет задать выравнивание полей структуры или класса.

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

где n - одно из следующих значений 1,2,4,8 или 16. Может отсутствовать.

Пример:

```
   struct pack(sizeof(long)) MyStruct
     {
      // члены структуры будут выровнены на границу 8 байт
     };
или
   struct MyStruct pack(sizeof(long))
     {
      // члены структуры будут выровнены на границу 8 байт
     };
```

По умолчанию для структур используется pack(1). Это означает, что в памяти члены структуры располагаются друг за другом и размер структуры равен сумме размеров её членов.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- простая структура без выравнивания
   struct Simple_Structure
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- объявим экземпляр простой структуры
   Simple_Structure s;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выведем размер каждого члена структуры
   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));
//--- убедимся, что размер POD-структуры равен сумме размеров её членов
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
   Результат:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=15
*/
  }
```

Выравнивание полей структуры может понадобится при обмене данными со сторонними библиотеками (*.DLL), в которых такое выравнивание применяется.

Покажем на примерах, как работает выравнивание. Возьмем структуру из четырех членов без выравнивания.

```
//--- простая структура без выравнивания
   struct Simple_Structure pack() // размер не указан, будет выравнивание на границу в
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
//--- объявим экземпляр простой структуры
   Simple_Structure s;
```

Поля структуры будут располагаться в памяти друг за другом, согласно порядку объявления и размеру типа. Размер структуры равен 15, смещение к полям структуры в массивах будет неопределённым.



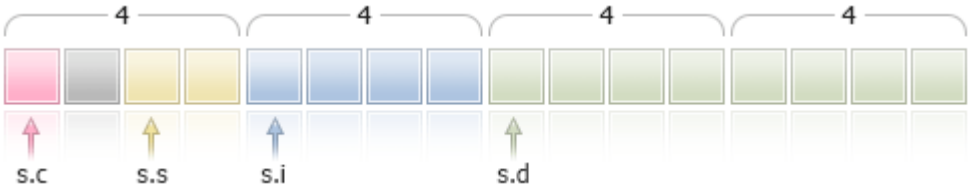
Объявим теперь эту же структуру с выравниванием в 4 байта и запустим код.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- простая структура с выравниванием в 4 байта
```

© 2000-2025, MetaQuotes Ltd.

```
   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
     };
   //--- объявим экземпляр простой структуры
   Simple_Structure s;
//--- выведем размер каждого члена структуры
   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));
//--- убедимся, что размер POD-структуры теперь не равен сумме размеров её членов
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
   Результат:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=16 // размер структуры изменился
*/
  }
```

Размер структуры изменился таким образом, чтобы все члены размером 4 байта или больше имели смещение от начала структуры кратное 4 байтам. Члены меньшего размера будут выравниваться на границу своего размера (например 2 для short). Вот как это выглядит, добавленный байт показан серым цветом.



В данном случае после члена **s.c** добавлен 1 байт, чтобы поле **s.s** (sizeof(short)==2) имело границу 2 байта (выравниваение для типа short).

Смещение к началу структуры в массиве также будет выравнено на границу 4 байт, т.е. для Simple_Structure arr[], адреса элементов a[0], a[1], a[n] будут кратными 4 байтам.

Рассмотрим еще две структуры, которые состоят из одинаковых типов с выравниванием на 4 байта, но при этом порядок следования членов отличается. В первой структуре члены располагаются по возрастанию размера типа.

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

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- простая структура с выравниванием на границу 4 байта
   struct CharShortInt pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(double)=4
     };
//--- объявим экземпляр простой структуры
   CharShortInt ch_sh_in;
//--- выведем размер каждого члена структуры
   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));

//--- убедимся, что размер POD-структуры равен сумме размеров её членов
   Print("sizeof(CharShortInt)=",sizeof(CharShortInt));
/*
   Результат:
   sizeof(ch_sh_in.c)=1
   sizeof(ch_sh_in.s)=2
   sizeof(ch_sh_in.i)=4
   sizeof(CharShortInt)=8
*/
  }
```

Как видим, размер структуры равен 8 и состоит из двух блоков по 4 байта. В первом блоке размещаются поля с типами char и short, во втором блоке находится поле с типом int.



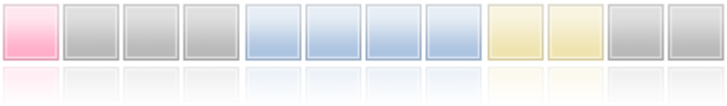
Теперь из первой структуры сделаем вторую, которая отличается только порядком следования полей - переставим член типа short в конец.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- простая структура с выравниванием на границу 4 байта
   struct CharIntShort pack(4)
     {
      char              c; // sizeof(char)=1
      int               i; // sizeof(double)=4
      short             s; // sizeof(short)=2
     };
//--- объявим экземпляр простой структуры
```

© 2000-2025, MetaQuotes Ltd.

```
   CharIntShort ch_in_sh;
//--- выведем размер каждого члена структуры
   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));
//--- убедимся, что размер POD-структуры равен сумме размеров её членов
   Print("sizeof(CharIntShort)=",sizeof(CharIntShort));
/*
   Результат:
   sizeof(ch_in_sh.c)=1
   sizeof(ch_in_sh.i)=4
   sizeof(ch_in_sh.s)=2
   sizeof(CharIntShort)=12
*/
  }
```

Хотя сам состав структуры практически не изменился, но изменение порядка членов привело к увеличению размера самой структуры.



При наследовании также необходимо учитывать выравнивание. Покажем на примере простой структуры Parent, которая имеет только один член типа char. Размер такой структуры без выравнивания равен 1.

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

Создаём дочерний класс Children с добавлением члена типа short (sizeof(short)=2).

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

В результате при установке выравнивания в 2 байта размер структуры будет равен 4, хотя размер самих членов в ней равен 3. В данном примере под родительский класс Parent будет выделено 2 байта, чтобы доступ к полю short дочернего класса был выравнен на 2 байта.

Знание того, как распределяется память под члены структуры необходимо, если MQL5-программа взаимодействует со сторонними данными посредством записи/чтения на уровне файлов или потоков.

В Стандартной библиотеке в каталоге MQL5\Include\WinAPI представлены функции для работы с функциями WinAPI. Эти функции используют структуры с заданным выравниванием для тех случаев, когда это требуется для работы с WinAPI.

© 2000-2025, MetaQuotes Ltd.

offsetof - это специальная команда, которая непосредственно связана в атрибутом pack. Она позволяет получить смещение члена от начала структуры.

```
//--- объявим переменную типа Children
   Children child;
//--- узнаем смещения от начала структуры
   Print("offsetof(Children,c)=",offsetof(Children,c));
   Print("offsetof(Children,s)=",offsetof(Children,s));
/*
   Результат:
   offsetof(Children,c)=0
   offsetof(Children,s)=2
*/
```

Спецификатор final

Наличие спецификатора final при объявлении структуры запрещает дальнейшее наследование от нее. Если структура такова, что нет необходимости вносить в нее дальнейшие изменения, или изменения не допустимы по соображениям безопасности, объявите ее со спецификатором final. При этом все члены структуры будут также неявно считаться final.

```
struct settings final
  {
  //--- тело структуры
  };

struct trade_settings : public settings
  {
  //--- тело структуры
  };
```

При попытке наследования от структуры с спецификатором final, как показано в примере выше, компилятор выдаст ошибку:

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

Классы

Классы имеют ряд отличий от структур:

- в объявлении используется ключевое слово class;
- по умолчанию все члены класса имеют спецификатор доступа private, если не указано иное. Члены-данные структуры по умолчанию имеют тип доступа public, если не указано иное;
- объекты классов всегда имеют таблицу виртуальных функций, даже если в классе не объявлено ни одной виртуальной функции. Структуры не могут иметь виртуальных функций;
- к объектам класса можно применять оператор new, к структурам этот оператор применять нельзя;
- классы могут наследоваться только от классов, структуры могут наследоваться только от структур.

© 2000-2025, MetaQuotes Ltd.

Классы и структуры могут иметь явный конструктор и деструктор. В случае если явно определен конструктор, инициализация переменной типа структуры или класса при помощи инициализирующей последовательности невозможна.

Пример:

```
struct trade_settings
  {
   double take;          // значения цены фиксации прибыли
   double stop;          // значение цены защитного стопа
   uchar  slippage;      // значение допустимого проскальзывания
   //--- конструктор
          trade_settings() { take=0.0; stop=0.0; slippage=5; }
   //--- деструктор
         ~trade_settings() { Print("Это конец"); }
  };
//--- компилятор выдаст ошибку с сообщением о невозможности инициализации
trade_settings my_set={0.0,0.0,5};
```

Конструкторы и деструкторы

Конструктор - это специальная функция, которая вызывается автоматически при создании объекта структуры или класса и обычно используется для инициализации членов класса. Далее мы будем говорить только о классах, при этом все сказанное относится и к структурам, если не оговорено иное. Имя конструктора должно совпадать с именем класса. Конструктор не имеет возвращаемого типа (можно указать тип void).

Определенные члены класса - строки, динамические массивы и объекты, требующие инициализации - в любом случае будут проинициализированы, независимо от наличия конструктора.

Каждый класс может иметь несколько конструкторов, отличающихся по количеству параметров и спискам инициализации. Конструктор, требующий указания параметров, называется параметрическим конструктором.

Конструктор, не имеющий параметров, называется **конструктором по умолчанию**. Если в классе не объявлен ни один конструктор, то компилятор сам создаст конструктор по умолчанию при компиляции.

```
//+------------------------------------------------------------------+
//| Класс для работы с датой                                         |
//+------------------------------------------------------------------+
class MyDateClass
  {
private:
   int               m_year;          // год
   int               m_month;         // месяц
   int               m_day;           // день месяца
   int               m_hour;          // час в сутках
   int               m_minute;        // минуты
   int               m_second;        // секунды
```

© 2000-2025, MetaQuotes Ltd.

```
public:
   //--- конструктор по умолчанию
                     MyDateClass(void);
   //--- конструктор с параметрами
                     MyDateClass(int h,int m,int s);
  };
```

Конструктор можно объявить в описании класса, а затем определить его тело. Например, вот так могут быть определены два конструктора класса MyDateClass:

```
//+------------------------------------------------------------------+
//| Конструктор по умолчанию                                         |
//+------------------------------------------------------------------+
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__);
  }
//+------------------------------------------------------------------+
//| Конструктор с параметрами                                        |
//+------------------------------------------------------------------+
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__);
  }
```

В конструкторе по умолчанию заполняются все члены класса с помощью функции TimeCurrent(), в конструкторе с параметрами заполняются только значения часа. Остальные члены класса (m_year, m_month и m_day) будут проинициализированы автоматически текущей датой.

© 2000-2025, MetaQuotes Ltd.

Конструктор по умолчанию имеет специальное назначение при инициализации массива объектов своего класса. Конструктор, все параметры которого имеют значения по умолчанию, **не является** конструктором по умолчанию. Покажем это на примере:

```
//+------------------------------------------------------------------+
//| Класс с конструктором по умолчанию                               |
//+------------------------------------------------------------------+
class CFoo
  {
   datetime          m_call_time;     // время последнего обращения к объекту
public:
   //--- конструктор с параметром, имеющем значение по умолчанию, не является конструк
                     CFoo(const datetime t=0){m_call_time=t;};
   //--- конструктор копирования
                     CFoo(const CFoo &foo){m_call_time=foo.m_call_time;};

   string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
// CFoo foo; // такой вариант использовать нельзя - конструктор по умолчанию не задан
//--- допустимые варианты создания объекта CFoo
   CFoo foo1(TimeCurrent());      // явный вызов параметрического конструктора
   CFoo foo2();                   // явный вызов параметрического конструктора с параме
   CFoo foo3=D'2009.09.09';       // неявный вызов параметрического конструктора
   CFoo foo40(foo1);              // явный вызов конструктора копирования
   CFoo foo41=foo1;               // неявный вызов конструктора копирования
   CFoo foo5;                     // явный вызов конструктора по умолчанию (если констр
                                  // то вызывается параметрический конструктор с параме
//--- допустимые варианты получения указателей CFoo
   CFoo *pfoo6=new CFoo();        // динамическое создание объекта и получение указател
   CFoo *pfoo7=new CFoo(TimeCurrent());// ещё один вариант динамического создания объе
   CFoo *pfoo8=GetPointer(foo1); // теперь pfoo8 указывает на объект foo1
   CFoo *pfoo9=pfoo7;             // pfoo9 и pfoo7 указывают на один и тот же объект
   // CFoo foo_array[3];          // такой вариант использовать нельзя - конструктор по
//--- выведем значения 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());
//--- удалим динамически созданные объекты
```

© 2000-2025, MetaQuotes Ltd.

```
   delete pfoo6;
   delete pfoo7;
   //delete pfoo8;  // удалять pfoo8 явно не нужно, так как он указывает на автоматиче
   //delete pfoo9;  // удалять pfoo9 явно не нужно, так как он указывает на тот же об
  }
```

Если раскомментировать в этом примере строки

```
   //CFoo foo_array[3];     // такой вариант использовать нельзя - конструктор по умол
```

или

```
   //CFoo foo_dyn_array[];  // такой вариант использовать нельзя - конструктор по умол
```

то компилятор выдаст на них ошибку "default constructor is not defined".

Если класс имеет конструктор, объявленный пользователем, то конструктор по умолчанию не будет сгенерирован компилятором. Это означает, что если в классе объявлен конструктор с параметрами, но не объявлен конструктор по умолчанию, то нельзя объявлять массивы объектов этого класса. Вот на таком скрипте компилятор сообщит об ошибке:

```
//+------------------------------------------------------------------+
//| Класс без конструктора по умолчанию                              |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
                     CFoo(string name) { m_name=name;}
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- при компиляции получим ошибку "default constructor is not defined"
   CFoo badFoo[5];
  }
```

В данном примере класс CFoo имеет объявленный параметрический конструктор - в таких случаях компилятор при компиляции не создает автоматически конструктор по умолчанию. В то же время при объявлении массива объектов предполагается, что все объекты должны быть созданы и инициализированы автоматически. При автоматической инициализации объекта требуется вызвать конструктор по умолчанию, но так как конструктор по умолчанию явно не объявлен и не сгенерирован автоматически компилятором, то создание такого объекта невозможно. Именно по этой причине компилятор выдает ошибку еще на этапе компиляции.

Существует специальный синтаксис для инициализации объекта с помощью конструктора. Инициализаторы конструктора (специальные конструкции для инициализации) для членов структуры или класса могут быть заданы в списке инициализации.

© 2000-2025, MetaQuotes Ltd.

Список инициализации - это список инициализаторов, разделенных запятыми, который идет после двоеточия за списком параметров конструктора и предшествует телу (идет перед открывающей фигурной скобкой). Есть несколько требований:

- списки инициализации можно использовать только в конструкторах;
- в списке инициализации нельзя инициализировать члены родителей;
- после списка инициализации должно идти определение (реализация) функции.

Покажем пример нескольких конструкторов для инициализации членов класса.

```
//+------------------------------------------------------------------+
//| Класс для хранения фамилии и имени персонажа                     |
//+------------------------------------------------------------------+
class CPerson
  {
   string            m_first_name;     // имя
   string            m_second_name;    // фамилия
public:
   //--- пустой конструктор по умолчанию
                     CPerson() {Print(__FUNCTION__);};
   //--- параметрический конструктор
                     CPerson(string full_name);
   //--- конструктор со списком инициализации
                     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);
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим ошибку "default constructor is not defined"
   CPerson people[5];
   CPerson Tom="Tom Sawyer";                        // Том Сойер
   CPerson Huck("Huckleberry","Finn");              // Гекльберри Финн
   CPerson *Pooh = new CPerson("Winnie","Pooh");   // Винни Пух
   //--- выведем значения
```

© 2000-2025, MetaQuotes Ltd.

```
   Tom.PrintName();
   Huck.PrintName();
   Pooh.PrintName();

   //--- удалим динамически созданный объект
   delete Pooh;
  }
```

В данном случае класс CPerson имеет три конструктора:

1. явный конструктор по умолчанию, который позволяет создавать массив объектов данного класса;
2. конструктор с одним параметром, который принимает в качестве параметра полное имя и разделяет его на имя и фамилию по найденному пробелу;
3. конструктор с двумя параметрами, который содержит список инициализации. Инициализаторы - m_second_name(surname) и m_first_name(name).

Обратите внимание, как инициализация с помощью списка заменила присваивание. Отдельные члены должны быть инициализированы как:

```
член_класса (список выражений)
```

В списке инициализации члены могут идти в любом порядке, но при этом все члены класса будут инициализироваться согласно порядку их объявления. Это означает, что в третьем конструкторе сначала будет инициализирован член m_first_name, так как он объявлен первым, и только после него будет инициализирован член m_second_name. Это необходимо учитывать в тех случаях, когда инициализация одних членов класса зависит от значений в других членах класса.

Если в базовом классе не объявлен конструктор по умолчанию и при этом объявлен один или несколько конструкторов с параметрами, то нужно обязательно вызвать один из конструкторов базового класса в списке инициализации. Он идет через запятую как обычные члены списка и будет вызван в первую очередь при инициализации объекта независимо от местоположения в списке инициализации.

```
//+------------------------------------------------------------------+
//| Базовый класс                                                    |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
   //--- конструктор со списком инициализации
                     CFoo(string name) : m_name(name) { Print(m_name);}
  };
//+------------------------------------------------------------------+
//| Потомок класса CFoo                                              |
//+------------------------------------------------------------------+
class CBar : CFoo
  {
   CFoo              m_member;       // член класса является объектом предка
public:
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- конструктор по умолчанию в списке инициализации вызывает конструктор предка
                     CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   CBar bar;
  }
```

В приведенном примере при создании объекта bar будет вызван конструктор по умолчанию CBar(), в котором сначала вызывается конструктор для предка CFoo, а затем конструктор для члена класса m_member.

Деструктор - это специальная функция, которая вызывается автоматически при уничтожении объекта класса. Имя деструктора записывается как имя класса с тильдой (~). Строки, динамические массивы и объекты, требующие деинициализации, в любом случае будут деинициализированы независимо от наличия деструктора. При наличии деструктора, эти действия будут произведены после вызова деструктора.

Деструкторы всегда являются виртуальными, независимо от того, объявлены они с ключевым слово virtual или нет.

Определение методов классов

Функции-методы класса могут быть определены как внутри класса, так и за пределами объявления класса. Если метод определяется внутри класса, то его тело следует непосредственно после объявления метода.

Пример:

```
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);       }
```

© 2000-2025, MetaQuotes Ltd.

```
   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[]);
  };
```

Функции с SetRightBorder(int border) по Draw() объявлены и определены прямо внутри класса CTetrisShape.

Конструктор CTetrisShape() и методы CheckDown(int& pad_array[]), CheckLeft(int& side_row[]) и CheckRight(int& side_row[]) только объявлены внутри класса, но пока не определены. Определения этих функций должны следовать далее по коду. Для того чтобы определить метод вне класса используется операция разрешения контекста, в качестве контекста используется имя класса.

Пример:

```
//+------------------------------------------------------------------+
//| Конструктор базового класса                                      |
//+------------------------------------------------------------------+
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;
  }
//+------------------------------------------------------------------+
//| Проверка возможности двигаться вниз (для палки и куба)          |
//+------------------------------------------------------------------+
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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Спецификаторы доступа public, protected и private

При разработке нового класса рекомендуется ограничивать доступ к членам извне. Для этих целей используются ключевые слова private или protected. Доступ в этом случае к сокрытым данным может осуществляться только из функций-методов этого же класса. Если использовано ключевое слово protected, то доступ к сокрытым данным может осуществляться и из методов классов - наследников этого класса. Точно таким же образом может ограничиваться доступ и к функциям-методам класса.

Если необходимо полностью открыть доступ к членам и/или методам класса, то используется ключевое слово public.

Пример:

```
class CTetrisField
  {
private:
   int               m_score;                              // счёт
   int               m_ypos;                               // текущее положение фигуры
   int               m_field[FIELD_HEIGHT][FIELD_WIDTH]; // матрица стакана
   int               m_rows[FIELD_HEIGHT];                 // нумерация рядов стакана
   int               m_last_row;                           // последний свободный ряд
   CTetrisShape     *m_shape;                              // тетрисная фигура
   bool              m_bover;                              // игра закончена
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();
  };
```

Любые члены и методы класса, объявленные после спецификатора **public:** (и до следующего спецификатора доступа), доступны при любом обращении программы к объекту этого класса. В данном примере это следующие члены: функции CTetrisField(), Init(), Deinit(), Down(), Left(), Right(), Rotate() и Drop().

Любые члены класса, объявленные после спецификатора доступа **private:** (и до следующего спецификатора доступа), доступны только функциям-членам этого класса. Спецификаторы доступа к элементам всегда заканчиваются двоеточием (:) и могут появляться в определении класса много раз.

Любые члены класса, объявленные после спецификатора доступа **protected:** (и до следующего спецификатора доступа), доступны только функциям-членам этого класса и функциям-членам

© 2000-2025, MetaQuotes Ltd.

наследников этого класса. При попытке обратиться к членам со спецификаторами private и protected извне будет получена ошибка на стадии компиляции. Пример:

```
class A
  {
protected:
   //--- оператор копирования доступен только внутри класса A и его наследников
   void operator=(const A &)
     {
     }
  };
class B
  {
   //--- обьявлен объект класса A
   A                 a;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- объявим две переменные типа B
   B b1, b2;
   //--- попытка скопировать один объект в другой
   b2=b1;
  }
```

При компиляции этого кода будет получено сообщение об ошибке — попытка вызвать удаленный оператор копирования:

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

И второй строкой ниже дается более подробное описание — оператор копирования в классе B был явно удален, так как вызывается недоступный оператор копирования класса A:

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

Доступ к членам базового класса может переопределяться при наследовании в производных классах.

Спецификатор delete

Спецификатор delete помечает функции-члены класса, которые нельзя использовать. Это означает, что если программа ссылается явно или неявно на такую функцию, то будет получена ошибка уже на этапе компиляции. Например, данный спецификатор позволяет сделать недоступными родительские методы в дочернем классе. Такого же результата можно добиться, если в родительском классе объявить функцию в приватной области (объявления в секции private). Использование delete в данном случае делает код более читаемым и управляемым на уровне наследников.

```
class A
  {
```

© 2000-2025, MetaQuotes Ltd.

```
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                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим переменную типа A
   A a;
   Print("a.GetValue()=", a.GetValue());
//--- попытка получить значение из переменной типа B
   B b;
   Print("b.GetValue()=", b.GetValue()); // компилятор выдаст ошибку на этой строке
  }
```

Сообщение компилятора:

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

С помощью спецификатора delete можно запретить автоматическое приведение типов или конструктор копий, который иначе пришлось бы также прятать в private-секции. Пример:

```
class A
  {
public:
   void              SetValue(double v) {value=v;}
   //--- запрещаем вызов с типом int
   void              SetValue(int) = delete;
   //--- запрещаем оператор копирования
   void              operator=(const A&) = delete;
private:
   double            value;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим две переменные типа A
   A a1, a2;
   a1.SetValue(3);       // ошибка!
```

© 2000-2025, MetaQuotes Ltd.

```
   a1.SetValue(3.14);    // OK
   a2=a1;                // ошибка!
  }
```

При попытке компиляции получим сообщения об ошибках:

```
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
```

Спецификатор final

Наличие спецификатора final при объявлении класса запрещает дальнейшее наследование от него. Если интерфейс класса таков, что нет необходимости вносить в него дальнейшие изменения, или изменения не допустимы по соображениям безопасности, объявите класс со спецификатором final. При этом все методы класса будут также неявно считаться final.

```
class CFoo final
  {
  //--- тело класса
  };

class CBar : public CFoo
  {
  //--- тело класса
  };
```

При попытке наследования от класса с спецификатором final, как показано в примере выше, компилятор выдаст ошибку:

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

Объединение (union)

Объединение - это особый тип данных, который состоит из нескольких переменных, разделяющих одну и ту же область памяти. Следовательно, объединение обеспечивает возможность интерпретации одной и той же последовательности битов двумя (или более) различными способами. Объявление объединения подобно объявлению структуры и начинается с ключевого слова union.

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

Но в отличие от структуры, разные члены объединения относятся к одному и тому же участку памяти. В данном примере объявлено объединение LongDouble, в котором значение типа long и значение типа double разделяют одну и ту же область памяти. Важно понимать - невозможно сделать так, чтобы это объединение хранило одновременно целочисленное значение *long* и вещественное *double* (как это было бы в структуре), поскольку переменные long_value и double_value накладываются (в памяти) друг на друга. Но зато MQL5-программа в любой момент

© 2000-2025, MetaQuotes Ltd.

может обрабатывать информацию, содержащуюся в этом объединении, как целочисленное значение (long) или как вещественное (double). Следовательно, объединение позволяет получить два (или больше) варианта представления одной и той же последовательности данных.

При объявлении объединения компилятор автоматически выделяет область памяти, достаточную для хранения в объединении переменных самого большого по объему типа. Для доступа к элементу объединения используется тот же синтаксис, как и для структур - оператор "точка".

```
union LongDouble
{
  long   long_value;
  double double_value;
};
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   LongDouble lb;
//--- получим недействительное число -nan(ind) и выведем его
   lb.double_value=MathArcsin(2.0);
   printf("1.  double=%f                integer=%I64X",lb.double_value,lb.long_value);
//--- наибольшее нормализованное число (DBL_MAX)
   lb.long_value=0x7FEFFFFFFFFFFFFF;
   printf("2.  double=%.16e  integer=%I64X",lb.double_value,lb.long_value);
//--- наименьшее положительное нормализованное (DBL_MIN)
   lb.long_value=0x0010000000000000;
   printf("3.  double=%.16e  integer=%.16I64X",lb.double_value,lb.long_value);
  }
/*  Результат выполнения
    1.  double=-nan(ind)                integer=FFF8000000000000
    2.  double=1.7976931348623157e+308  integer=7FEFFFFFFFFFFFFF
    3.  double=2.2250738585072014e-308  integer=0010000000000000
*/
```

Поскольку объединения позволяют программе интерпретировать одни и те же данные в памяти по-разному, они часто используются в случаях, когда требуется необычное преобразование типов.

Объединения не могут участвовать в наследовании, а также они не могут иметь статических членов по определению. В остальном *union* ведёт себя как структура, у которой все члены имеют нулевое смещение. При этом членами объединения не могут быть следующие типы:

- динамические массивы
- строки
- указатели на объекты и функции
- объекты классов
- объекты структур, имеющие конструкторы или деструкторы
- объекты структур, имеющие в себе члены из пунктов 1-5

© 2000-2025, MetaQuotes Ltd.

Также как и классы, объединение может иметь конструкторы и деструкторы, а также и методы. По умолчанию члены объединения имеют тип доступа public, для создания закрытых элементов необходимо использовать ключевое слово private. Все эти возможности представлены в примере, который показывает как преобразовать цвет, имеющий тип color, в представление ARGB, как это делает функция ColorToARGB().

```
//+------------------------------------------------------------------+
//| Объединение для конвертации color(BGR) в представление ARGB      |
//+------------------------------------------------------------------+
union ARGB
  {
   uchar             argb[4];
   color             clr;
   //--- конструкторы
                     ARGB(color col,uchar a=0){Color(col,a);};
                    ~ARGB(){};
   //--- публичные методы
public:
   uchar   Alpha(){return(argb[3]);};
   void    Alpha(const uchar alpha){argb[3]=alpha;};
   color   Color(){ return(color(clr));};
   //--- закрытые методы
private:
   //+------------------------------------------------------------------+
   //| установка цвета и значения альфа-канала                          |
   //+------------------------------------------------------------------+
   void    Color(color col,uchar alpha)
     {
      //--- установим цвет в член clr
      clr=col;
      //--- установим значение компонента Alpha - уровня непрозрачности
      argb[3]=alpha;
      //--- переставим местами байты компонент R и B (Red и Blue)
      uchar t=argb[0];argb[0]=argb[2];argb[2]=t;
     };
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- значение 0x55 означает 55/255=21.6 % (0% - полностью прозрачный)
   uchar alpha=0x55;
//--- тип color имеет представление 0x00BBGGRR
   color test_color=clrDarkOrange;
//--- сюда будем принимать значения байтов из объединения ARGB
   uchar argb[];
   PrintFormat("0x%.8X - так выглядит тип color для %s, BGR=(%s)",
               test_color,ColorToString(test_color,true),ColorToString(test_color));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- тип ARGB представлен как 0x00RRGGBB, переставлены местами RR и BB компоненты
   ARGB argb_color(test_color);
//--- скопируем массив байтов
   ArrayCopy(argb,argb_color.argb);
//--- посмотрим как выглядит в представлении ARGB
   PrintFormat("0x%.8X - представление ARGB с альфа-каналом=0x%.2x, ARGB=(%d,%d,%d,%d)
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- добавим значение непрозрачности
   argb_color.Alpha(alpha);
//--- попробуем вывести ARGB как тип color
   Print("ARGB как color=(",argb_color.clr,")  альфа-канал=",argb_color.Alpha());
//--- скопируем массив байтов
   ArrayCopy(argb,argb_color.argb);
//--- а вот как выглядит в представлении ARGB
   PrintFormat("0x%.8X - представление ARGB с альфа-каналом=0x%.2x, ARGB=(%d,%d,%d,%d)
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- сверим с тем, что выдает функция ColorToARGB()
   PrintFormat("0x%.8X - результат ColorToARGB(%s,0x%.2x)",ColorToARGB(test_color,alph
               ColorToString(test_color,true),alpha);
  }
/* Результат выполнения
   0x00008CFF - так выглядит тип color для clrDarkOrange, BGR=(255,140,0)
   0x00FF8C00 - представление ARGB с альфа-каналом=0x00, ARGB=(0,255,140,0)
   ARGB как color=(0,140,255)  альфа-канал=85
   0x55FF8C00 - представление ARGB с альфа-каналом=0x55, ARGB=(85,255,140,0)
   0x55FF8C00 - результат ColorToARGB(clrDarkOrange,0x55)
*/
```

Интерфейсы

Интерфейс предназначен для определения определённого функционала, который класс впоследствии может реализовывать. Фактически, это класс, который не может содержать члены и не может иметь конструктор и/или деструктор. Все объявленные в интерфейсе методы являются чисто виртуальными, даже без явного определения.

Определяется интерфейс с помощью ключевого слова interface, как показано в примере:

```
//--- базовый интерфейс для описания животных
interface IAnimal
  {
//--- методы интерфейса по умолчанию имеют public-доступ
   void Sound();  // звук, который издает животное
  };
//+------------------------------------------------------------------+
//|  класс CCat наследуется от интерфейса IAnimal                    |
//+------------------------------------------------------------------+
class CCat : public IAnimal
  {
public:
```

© 2000-2025, MetaQuotes Ltd.

```
                     CCat() { Print("Cat was born"); }
                    ~CCat() { Print("Cat is dead");  }
   //--- реализуем метод Sound интерфейса IAnimal
   void Sound(){ Print("meou"); }
  };
//+------------------------------------------------------------------+
//|  класс CDog наследуется от интерфейса IAnimal                    |
//+------------------------------------------------------------------+
class CDog : public IAnimal
  {
public:
                     CDog() { Print("Dog was born"); }
                    ~CDog() { Print("Dog is dead");  }
   //--- реализуем метод Sound интерфейса IAnimal
   void Sound(){ Print("guaf"); }
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- массив указателей на объекты типа IAnimal
   IAnimal *animals[2];
//--- породим потомков IAnimal и сохраним указатели на них в массив
   animals[0]=new CCat;
   animals[1]=new CDog;
//--- вызовем метод Sound() базового интерфейса IAnimal для каждого потомка
   for(int i=0;i<ArraySize(animals);++i)
      animals[i].Sound();
//--- удалим объекты
   for(int i=0;i<ArraySize(animals);++i)
      delete animals[i];
//--- результат выполнения
/*
   Cat was born
   Dog was born
   meou
   guaf
   Cat is dead
   Dog is dead
*/
  }
```

Как и в случае абстрактных классов, нельзя создавать объект интерфейса без наследования. Интерфейс может наследоваться только от других интерфейсов и может выступать предком для класса. При этом интерфейс всегда имеет публичную видимость.

Интерфейс нельзя объявить внутри объявления класса или структуры, но при этом указатель на интерфейс можно сохранить в переменную типа void *. Вообще говоря, в переменную типа void *

© 2000-2025, MetaQuotes Ltd.

можно сохранить указатель на объект любого класса. Для того чтобы преобразовать указатель void * в указатель на объект конкретного класса, необходимо использовать оператор dynamic_cast. В случае, когда преобразование невозможно, результатом операции dynamic_cast будет NULL.

Смотри также

Объектно-ориентированное программирование

© 2000-2025, MetaQuotes Ltd.

Объект динамического массива

Динамические массивы

Допускается объявление не более чем 4-мерного массива. При объявлении динамического массива (массива с неуказанным значением в первой паре квадратных скобок) компилятор автоматически создает переменную указанной выше структуры (объект динамического массива) и обеспечивает код для правильной инициализации.

Динамические массивы автоматически освобождаются при выходе за пределы области видимости блока, в котором они объявлены.

Пример:

```
double matrix[][10][20]; // 3-мерный динамический массив
ArrayResize(matrix,5);   // задали размер первого измерения
```

Статические массивы

При явном указании всех значимых размерностей массива компилятор заранее распределяет необходимый размер памяти. Такой массив называется статическим. Тем не менее, компилятор дополнительно распределяет память под объект динамического массива, который (объект) связан с заранее распределенным статическим буфером (областью памяти для хранения массива).

Создание объекта динамического массива обусловлено возможной необходимостью передавать данный статический массив в качестве параметра в какую-либо функцию.

Примеры:

```
double stat_array[5]; // 1-мерный статический массив
some_function(stat_array);
...
bool some_function(double& array[])
  {
   if(ArrayResize(array,100)<0)
      return(false);
   ...
   return(true);
  }
```

Массивы в составе структур

При объявлении статического массива в качестве члена структуры объект динамического массива не создается. Это сделано для совместимости структур данных, используемых в Windows API.

Однако статические массивы, объявленные в качестве членов структур, также можно передавать в MQL5-функции. В этом случае при передаче параметра будет создан временный объект динамического массива, связанный со статическим массивом - членом структуры.

Смотри также

© 2000-2025, MetaQuotes Ltd.

Операции с массивами, Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Матрицы и векторы

vector — это специальный тип данных в MQL5 для работы с векторами, который представляет из себя одномерный массив типа double. Вектор является одним из основополагающих понятий линейной алгебры и активно применяется во многих областях науки — в физике, геометрии и других. Векторы используются при решении систем линейных уравнений, в 3D-графике и т.д. Над векторами определены операции сложения и умножения, а также введено понятие "Норма" для получения длины или расстояния между векторами. В программировании векторы обычно представлены массивами однородных элементов, над которыми может быть не задано обычных векторных операций, то есть массивы нельзя складывать друг с другом, умножать друг на друга, а также нет понятия нормы.

Векторы могут быть представлены как вектор-столбец или как вектор-строка при работе с матрицами. Помимо этого в линейной алгебре для векторов используются такие понятия, как ковариантность и контравариантность. Для целей написания кода на MQL5 эти различия не имеют значения, так как только сам программист решает, что из себя представляет каждый объект типа vector. Например, это может быть вектор поворота, перемещения, сжатия и так далее, если речь идет о 3D-графике.

Вообще говоря, с точки зрения линейной алгебры число также является вектором, только в одномерном пространстве. В то же время вектор можно рассматривать как частный случай матрицы.

matrix — это еще один специальный тип данных в MQL5 для представления матриц, который фактически является двумерным массивом типа double. Векторы и матрицы введены в язык MQL5 для более простой работы с определенными типами наборов данных. Они дают возможность программистам использовать все преимущества линейной алгебры в наиболее простой и приближенной к математике форме. Матрицы широко применяются в математике для компактной записи систем линейных алгебраических или дифференциальных уравнений. В этом случае количество строк матрицы соответствует числу уравнений, а количество столбцов — количеству неизвестных. В результате решение систем линейных уравнений сводится к операциям над матрицами.

Существуют следующие типы данных:

- matrix — матрица, содержащая элементы типа double
- matrixf — матрица, содержащая элементы типа float
- matrixc — матрица, содержащая элементы типа complex.
- vector — вектор, содержащий элементы типа double
- vectorf — вектор, содержащий элементы типа float
- vectorc — вектор, содержащий элементы типа complex

Для применения в шаблонных функциях можно использовать запись matrix<double>, matrix<float>, matrix<complex>, vector<double>, vector<float>, vector<complex> вместо соответствующих типов.

Для матрицы определены следующие алгебраические операции:

- сложение матриц, имеющих одинаковый размер;
- умножение матриц подходящего размера, при этом число столбцов матрицы слева должно соответствовать числу строк матрицы справа;

© 2000-2025, MetaQuotes Ltd.

- умножение матрицы на вектор-столбец и умножение вектор-строки на матрицу по правилу матричного умножения. То есть вектор является в этом смысле частным случаем матрицы;
- умножение матрицы на число, то есть на скаляр.

В математике рассматривается множество различных типов и видов матриц. Например, единичная, симметричная, кососимметричная, верхнетреугольная (нижнетреугольная) и другие матрицы. Особое значение в теории матриц занимают всевозможные нормальные формы, то есть канонический вид, к которому можно привести матрицу заменой координат. На практике используются такие нормальные формы, которые обладают дополнительными свойствами, например, устойчивостью.

Использование векторов и матриц, а точнее, специальных методов, которыми обладают эти типы данных, позволяет писать более простой, короткий и понятный код, который приближен к математической записи. Это избавляет программиста от необходимости создавать вложенные циклы и помнить о правильной индексации массивов, которые участвуют в вычислении. Таким образом повышается надежность и скорость разработки сложных программ.

Список методов, реализованных в matrix и vector

Типы matrix и vector имеют встроенные методы, которые соответствуют аналогам библиотеки NumPy. Это позволит перенести алгоритмы и коды с языка Python на MQL5 с минимальными затратами. Таким образом, множество задач по обработке данных, решению математических уравнений, нейросетям и машинному обучению можно решать с использованием готовых наработок и библиотек языка Python.

Метод matrix/vector	Аналог в NumPy	Описание
void matrix.Eye(const int rows, const int cols, const int ndiag=0)	eye	Создает матрицу с единицами по диагонали и нулями в остальных местах
void matrix.Identity(const int rows)	identity	Создает квадратную матрицу с единицами на главной диагонали
void matrix.Ones(const int rows, const int cols)	ones	Создает новую матрицу по числу строк и столбцов, заполненную единицами
void matrix.Zeros(const int rows, const int cols)	zeros	Создает новую матрицу по числу строк и столбцов, заполненную нулями.
void matrix.Full(const int rows, const int cols, const scalar value)	full	Создает новую матрицу по числу строк и столбцов, заполненную скалярным значением
void matrix.Copy(const matrix a) void vector.Copy(const vector v)	copy	Создает копию данной матрицы
void matrix.FromBuffer(const int rows, const int cols, const scalar	frombuffer	Создает матрицу из одномерного массива

© 2000-2025, MetaQuotes Ltd.

Метод matrix/vector	Аналог в NumPy	Описание
array[], const int count=-1, const int offset=0)		
void matrix.FromFile(const int rows, condt int cols, const int file_handle, const int count=-1, const int offset=0)	fromfile	Создает матрицу из данных в текстовом или двоичном файле
void vector.FromString(const string source, const string sep=" ")	fromstring	Создает вектор на основе текстовых данных из строки
void vector.Arange(const scalar start, const scalar stop, const scalar step=1)	arange	Создает вектор с равномерно распределенными значениями в заданном интервале
void matrix.Diag(const vector v, const int ndiag=0)	diag	Извлекает диагональ или создает диагональную матрицу
void matrix.Tri(const int rows, const int cols, const int ndiag=0)	tri	Создает матрицу с единицами на указанной диагонали и ниже и с нулями в остальных местах
void matrix.Tril(const int rows, const int cols, const scalar array[], const int ndiag=0)	tril	Возвращает копию матрицы с обнуленными элементами над k-й диагональю
void matrix.Triu(const int rows, const int cols, const scalar array[], const int ndiag=0)	triu	Возвращает копию матрицы с обнуленными элементами ниже k-й диагонали
void matrix.Vander(const vector v, const int cols=-1, const bool increasing=false)	vander	Создает матрицу Вандермонда
vector matrix.Row(const unsigned nrow)		Возвращает вектор-строку
vector matrix.Col(const unsigned ncol)		Возвращает вектор-столбец
unsigned matrix.Rows()		Возвращает количество строк в матрице
unsigned matrix.Cols()		Возвращает количество столбцов в матрице
void matrix.Init()		Инициализирует матрицу
matrix matrix.Transpose()	transpose	Меняет местами оси матрицы, возвращает модифицированную матрицу
matrix matrix.Dot(const matrix b)	dot	Произведение двух матриц
matrix matrix.Inner(const matrix b)	inner	Внутренний продукт двух матриц

© 2000-2025, MetaQuotes Ltd.

Метод matrix/vector	Аналог в NumPy	Описание
matrix matrix.Outer(const matrix b)	outer	Внешний продукт двух матриц
matrix matrix.MatMul(const matrix b)	matmul	Матричное произведение двух матриц
matrix matrix.MatrixPower(const int power)	matrix_power	Возводит квадратную матрицу в (целую) степень n
matrix matrix.Kron(const matrix b)	kron	Возвращает произведение Кронекера двух матриц
bool matrix.Cholesky(matrix& L)	cholesky	Возвращает разложение Холецкого
bool matrix.QR(matrix& Q, matrix& R)	qr	Вычисляет qr-разложение матрицы
bool matrix.SVD(matrix& U, matrix& V, vector& singular_values)	svd	Вычисляет SVD разложение
bool matrix.Eig(matrix& eigen_vectors, vector& eigen_values)	eig	Вычисляет собственные значения и правые собственные векторы квадратной матрицы
bool matrix.EigH(matrix& eigen_vectors, vector& eigen_values)	eigh	Возвращает собственные значения и собственные векторы Эрмитовой матрицы
bool matrix.EigVals(vector& eigen_values)	eigvals	Вычисляет собственные значения общей матрицы
bool matrix.EigValsH(vector& eigen_values)	eigvalsh	Вычисляет собственные значения Эрмитовой матрицы
bool matrix.LU(matrix& L, matrix& U)		Выполняет LU-декомпозицию матрицы как произведения нижней треугольной матрицы и верхней треугольной матрицы
bool matrix.LUP(matrix& L, matrix& U, matrix& P)		Выполняет LUP-декомпозицию с частичным поворотом, которая является LU-факторизацией с перестановками строк: PA = LU
double matrix.Norm(const norm)	norm	Возвращает норму матрицы или вектора
double matrix.Cond (const norm)	cond	Вычисляет число обусловленности матрицы
vector matrix.Spectrum()		Вычисляет спектр матрицы как набор её собственных значений от произведения AT*A
double matrix.Det()	det	Вычисляет детерминант матрицы

© 2000-2025, MetaQuotes Ltd.

Метод matrix/vector	Аналог в NumPy	Описание
int matrix.Rank()	matrix_rank	Возвращает матричный ранг массива с помощью метода Гаусса
int matrix.SLogDet(int& sign)	slogdet	Вычисляет знак и логарифм детерминанта матрицы
double matrix.Trace()	trace	Возвращает сумму по диагоналям матрицы
vector matrix.Solve(const vector b)	solve	Решает линейное матричное уравнение или систему линейных алгебраических уравнений
vector matrix.LstSq(const vector b)	lstsq	Решает систему линейных алгебраических уравнений приблизительно (для неквадратных или вырожденных матриц)
matrix matrix.Inv()	inv	Вычисляет (мультипликативную) обратную матрицу
matrix matrix.PInv()	pinv	Вычисляет псевдообратную матрицу методом Мура-Пенроуза
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)		Сравнивает элементы двух матриц/векторов с указанной точностью
double matrix.Flat(const ulong index) bool matrix.Flat(const ulong index,const double value)	flat	Позволяет адресоваться к элементу матрицы через один индекс, а не через два
double vector.ArgMax() double matrix.ArgMax() vector matrix.ArgMax(const int axis)	argmax	Возвращает индекс максимального значения
double vector.ArgMin() double matrix.ArgMin() vector matrix.ArgMin(const int axis)	argmin	Возвращает индекс минимального значения
double vector.Max() double matrix.Max() vector matrix.Max(const int axis)	max	Возвращает максимальное значение в матрице/векторе
double vector.Mean() double matrix.Mean()	mean	Вычисляет среднее арифметическое значений

© 2000-2025, MetaQuotes Ltd.

Метод matrix/vector	Аналог в NumPy	Описание
vector matrix.Mean(const int axis)		элементов
double vector.Min() double matrix.Min() vector matrix.Min(const int axis)	min	Возвращает минимальное значение в матрице/векторе
double vector.Sum() double matrix.Sum() vector matrix.Sum(const int axis)	sum	Выполняет суммирование элементов матрицы/вектора, которое так же может выполняться по указанной оси (осям).
void vector.Clip(const double min_value,const double max_value) void matrix.Clip(const double min_value,const double max_value)	clip	Ограничивает элементы матрицы/вектора указанным интервалом допустимых значений
vector vector.CumProd() vector matrix.CumProd() matrix matrix.CumProd(const int axis)	cumprod	Возвращает кумулятивное (накапливаемое) произведение элементов массива/вектора, в том числе и по заданной оси
vector vector.CumSum() vector matrix.CumSum() matrix matrix.CumSum(const int axis)	cumsum	Возвращает кумулятивную (накапливаемую) сумму элементов матрицы/вектора, в том числе и по заданной оси
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	Вычисляет произведение элементов матрицы/вектора, которое так же может выполняться по указанной оси
void matrix.Reshape(const ulong rows, const ulong cols)	reshape	Изменяет форму матрицы без изменения его данных
void matrix.Resize(const ulong rows,const ulong cols)	resize	Возвращает новую матрицу с указанной формой
bool matrix.SwapRows(const ulong row1, const ulong row2)		Обменивает местами строки в матрице
bool matrix.SwapCols(const ulong col1, const ulong col2)		Обменивает местами столбцы в матрице
double vector.Ptp() double matrix.Ptp() vector matrix.Ptp(const int axis)	ptp	Возвращает диапазон значений матрицы/вектора или указанной оси матрицы и эквивалентна результату Max() - Min()
double vector.Percentile(const int percent)	percentile	Вычисляет указанный процентиль (перцентиль) значений элементов матрицы/вектора или элементов

© 2000-2025, MetaQuotes Ltd.

Метод matrix/vector	Аналог в NumPy	Описание
double matrix.Percentile(const int percent) vector matrix.Percentile(const int percent,const int axis)		вдоль указанной оси. Допустимые значения параметра percent находятся в интервале [0, 100]
double vector.Quantile(const int percent) double matrix.Quantile(const int percent) vector matrix.Quantile(const int percent,const int axis)	quantile	Вычисляет указанный квантиль значений элементов матрицы/вектора или элементов вдоль указанной оси. Диапазон значений параметра percent принимает значения в интервале [0, 1]
double vector.Median() double matrix.Median() vector matrix.Median(const int axis)	median	Вычисляет медиану элементов матрицы/векторы. Медиана — это такое значение, что ровно половина элементов матрицы/вектора окажется меньше него, а другая больше.
double vector.Average() double matrix.Average() vector matrix.Average(const int axis)	average	Вычисляет среднее арифметическое взвешенное значений элементов матрицы/вектора. Сумма весов в знаменателе не может быть равна 0, но некоторые веса могут быть равны 0
double vector.Std() double matrix.Std() vector matrix.Std(const int axis)	std	Вычисляет среднеквадратичное (стандартное) отклонение значений элементов матрицы/вектора или указанной оси
double vector.Var() double matrix.Var() vector matrix.Var(const int axis)	var	Вычисляет дисперсию значений элементов матрицы/вектора
double vector.CorrCoef(const vector& v) matrix matrix.CorrCoef()	corrcoef	Вычисляет коэффициент корреляции Пирсона (линейный коэффициент корреляции). Коэффициент корреляции находится в интервале [-1, 1]
vector vector.Correlate(const vector& v,enum mode)	correlate	Вычисляет значения взаимнокорреляционной функции (кросс-корреляцию) двух векторов. Параметр mode определяет режим вычисления линейной свертки
vector vector.Convolve(const vector& v, enum mode)	convolve	Возвращает дискретную линейную свертку двух векторов последовательностей. Параметр

© 2000-2025, MetaQuotes Ltd.

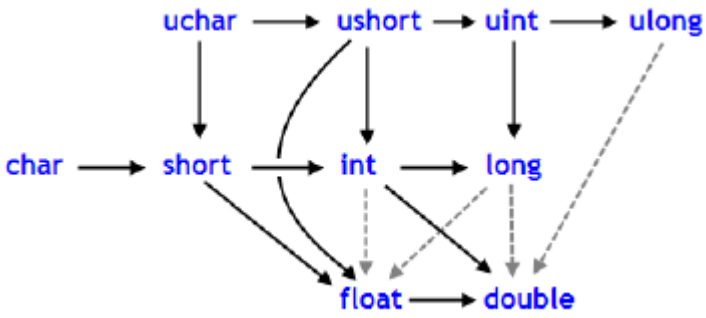
Метод matrix/vector	Аналог в NumPy	Описание
		mode определяет режим вычисления линейной свертки
matrix matrix.Cov() matrix vector.Cov(const vector& v); (результатная матрица 2 x 2)	cov	Вычисляет ковариационную матрицу. Ковариация двух выборок (двух случайных величин) — это мера их линейной зависимости

© 2000-2025, MetaQuotes Ltd.

Приведение типов

Преобразование числовых типов

Часто возникает необходимость преобразовать один числовой тип в другой. Не каждый числовой тип допустимо преобразовать в другой, допустимые преобразования в MQL5 показаны на схеме:



Сплошные линии со стрелками обозначают преобразования, которые выполняются без потери информации. Вместо типа char может выступать тип bool (оба занимают в памяти 1 байт), вместо типа int можно использовать тип color (по 4 байта), а вместо типа long допустим тип datetime (занимают по 8 байт). Четыре штриховые линии серого цвета, также со стрелками, означают преобразования, при которых может произойти потеря точности. Например, количество цифр в целом числе 123456789 (int) превышает количество цифр, которое может быть представлено типом float.

```
   int n=123456789;
   float f=n;     // содержимое f равно 1.234567892E8
   Print("n = ",n,"   f = ",f);
   // результат n= 123456789    f= 123456792.00000
```

Число, преобразованное в тип float, имеет тот же порядок, но несколько меньшую точность. Преобразования, обратные черным стрелкам, осуществляется с возможной потерей информацией. Преобразования между char и uchar, short и ushort, int и uint, long и ulong (имеются ввиду преобразования в обе стороны), могут привести к потере информации.

В результате преобразования значения с плавающей точкой к целому типу дробная часть числа всегда отбрасывается. Если нужно округлить число с плавающей точкой до ближайшего целого числа (что во многих случаях является более полезным), необходимо использовать функцию MathRound().

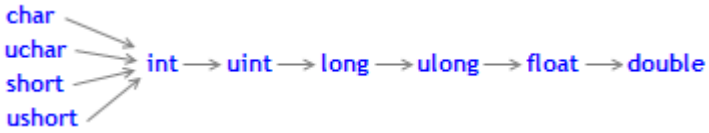
Пример:

```
//--- ускорение свободного падения
   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);
/*
   Результат:
   round_g = 9
```

© 2000-2025, MetaQuotes Ltd.

```
   math_round_g = 10
*/
```

Если два значения объединяются бинарным оператором, то перед выполнением операции операнд младшего типа преобразовывается к более старшему типу в соответствии с приоритетом, указанным на схеме:



Типы данных char, uchar, short и ushort в операциях безусловно приводятся к типу int.

Примеры:

```
   char   c1=3;
//--- первый пример
   double d2=c1/2+0.3;
   Print("c1/2+0.3 = ",d2);
// Результат:   c1/2+0.3 = 1.3

//--- второй пример
   d2=c1/2.0+0.3;
   Print("c1/2.0+0.3 = ",d2);
// Результат:   c1/2.0+0.3 = 1.8
```

Вычисляемое выражение состоит из двух операций. В первом примере переменная c1 типа char преобразуется ко временной переменной типа int, так как второй операнд в операции деления, константа 2, имеет более старший тип int. В результате целочисленного деления 3/2 получается значение 1, которое имеет тип int.

Во второй операции первого примера вторым операндом выступает константа 0.3, которая имеет тип double, поэтому результат первой операции преобразуется во временную переменную типа double со значением 1.0.

Во втором примере переменная c1 типа char преобразуется ко временной переменной типа double, так как второй операнд в операции деления, константа 2.0, имеет тип double; дальнейших преобразований не производится.

Приведение числовых типов

В выражениях языка MQL5 можно использовать как явное, так и неявное приведение типов. Явное преобразование типов записывается следующим образом:

```
var_1 = (тип)var_2;
```

В качестве переменной var_2 может быть выражение или результат выполнения функции. Допускается также функциональная запись явного приведения типов:

```
var_1 = тип(var_2);
```

© 2000-2025, MetaQuotes Ltd.

Рассмотрим явное преобразование на основании первого примера.

```
//--- третий пример
   double d2=(double)c1/2+0.3;
   Print("(double)c1/2+0.3 = ",d2);
// Результат:   (double)c1/2+0.3 = 1.80000000
```

Перед выполнением операции деления переменная c1 явно приводится к типу double. Теперь уже целочисленная константа 2 приводится к значению 2.0 типа double, так как в результате преобразования первый операнд получил тип double. Фактически, явное преобразование типов является одноместной операцией.

Кроме того, при попытке приведения типов результат может выйти за пределы допустимого диапазона. В этом случае произойдет усечение. Например:

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

Перед выполнением операций (кроме операций присваивания) происходит преобразование в тип, имеющий наибольший приоритет, а перед операциями присваивания - в целевой тип.

Примеры:

```
   int    i=1/2;          // приведения типов нет, результат: 0
   Print("i = 1/2  ",i);

   int k=1/2.0;           // выражение приводится к типу double,
   Print("k = 1/2  ",k);  // затем приводится к целевому типу int, результат: 0

   double d=1.0/2.0;      // приведения типов нет, результат: 0.5
   Print("d = 1/2.0; ",d);

   double e=1/2.0;        // выражение приводится к типу double,
   Print("e = 1/2.0; ",e);// который совпадает с целевым типом, результат: 0.5

   double x=1/2;          // выражение типа int приводится к целевому типу double,
   Print("x = 1/2; ",x);  // результат: 0.0
```

При преобразовании типа long/ulong в double может произойти потеря точности: если целое больше 9223372036854774784 или меньше -9223372036854774784.

```
void OnStart()
  {
   long l_max=LONG_MAX;
   long l_min=LONG_MIN+1;
//--- найдем максимальное целое, которое не теряет точности при приведении к double
   while(l_max!=long((double)l_max))
      l_max--;
//--- найдем минимальное целое, которое не теряет точности при приведении к double
   while(l_min!=long((double)l_min))
```

© 2000-2025, MetaQuotes Ltd.

```
      l_min++;
//--- теперь выведем найденный интервал для целых чисел
   PrintFormat("При приведении целого числа к double оно должно "
               "быть в интервале [%I64d, %I64d]",l_min,l_max);
//--- теперь посмотрим, что произойдет, если число выходит за этот интервал
   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)));
//--- получим такой вывод
// При приведении целого числа к double оно должно быть в интервале [-922337203685477
// 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_
  }
```

Приведения для типа string

Тип string имеет самый высокий приоритет среди простых типов. Поэтому, если в операции один из операндов имеет тип string, то другой операнд будет приведен к типу string автоматически. Следует иметь ввиду, что для типа string допустима единственная двуместная операция сложения. Допустимо явное приведение переменной типа string к любому числовому типу.

Примеры:

```
   string s1=1.0/8;              // выражение приводится к типу double,
   Print("s1 = 1.0/8; ",s1);     // затем к целевому типу string,
// результат:"0.12500000"(строка, содержащая 10 символов)

   string s2=NULL;               // деинициализация строки
   Print("s2 = NULL; ",s2);      // результат: пустая строка
   string s3="Ticket N"+12345;   // выражение приводится к типу string
   Print("s = \"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));
```

Приведение типов указателей базовых классов к указателям производных классов

© 2000-2025, MetaQuotes Ltd.

Объекты открыто порожденного класса могут также рассматриваться как объекты соответствующего ему базового класса. Это ведет к некоторым интересным следствиям. Например, вопреки тому факту, что объекты различных классов, порожденных одним базовым классом, могут существенно отличаться друг от друга, мы можем создать их связанный список (List), поскольку мы рассматриваем их как объекты базового типа. Но обратное неверно: объекты базового класса не являются автоматически объектами производного класса.

Можно использовать явное приведение типов для преобразования указателей базового класса в указатели производного класса. Но необходимо быть полностью уверенным в допустимости такого преобразования, так как в противном случае возникнет критическая ошибка времени выполнения и mql5-программа будет остановлена.

Динамическое приведение типов с помощью оператора dynamic_cast

Существует возможность динамического приведения типов с помощью оператора dynamic_cast, который может быть применён только к указателям классов. При этом проверка корректности типов производится в момент выполнения программы. Это означает, что при использовании оператора dynamic_cast компилятор не производит проверку типа данных, используемого для приведения. В случае, если осуществляется преобразование указателя к типу данных, который не является фактическим типом объекта, результатом будет значение NULL.

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

Параметр *type-id* в угловых скобках должен быть указателем на ранее определённый тип класса. Тип операнда *expression* (в отличии от C++) может быть любым, кроме void.

Пример:

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

void OnStart()
  {
   CBar bar;
//--- динамическое приведение типа указателя *bar к указателю *foo разрешено
   CFoo *foo = dynamic_cast<CFoo *>(&bar); // критической ошибки выполнения не возник
   Print(foo);                             // foo=NULL
//--- попытка явного приведения ссылки объекта типа Bar к объекту типа Foo запрещено
   foo=(CFoo *)&bar;                       // возникнет критическая ошибка выполнения
   Print(foo);                             // эта строка не будет выполнена
  }
```

Смотри также

Типы данных

© 2000-2025, MetaQuotes Ltd.

Тип void и константа NULL

Синтаксически тип void является фундаментальным типом наравне с типами char, uchar, bool, short, ushort, int, uint, color, long, ulong, datetime, float, double и string. Этот тип используется либо для указания того, что функция не возвращает значения, либо в качестве параметра функции обозначает отсутствие параметров.

Предопределенная константная переменная **NULL** имеет тип *void*. Она может быть присвоена переменным любых других фундаментальных типов без преобразования. Также допускается сравнение переменных фундаментальных типов со значением **NULL**

Пример:

```
//--- если строка не инициализирована, то присвоим ей наше предопределенное значение
if(some_string==NULL) some_string="empty";
```

Также **NULL** можно сравнивать с указателями на объекты, созданные при помощи оператора new.

Смотри также

Переменные, Функции

© 2000-2025, MetaQuotes Ltd.

Пользовательские типы

Ключевое слово typedef в языке C++ позволяет создавать пользовательские типы данных - для этого достаточно определить новое имя типа данных для уже существующего типа данных. При этом сам новый тип данных не создается, а лишь определяется новое имя для уже существующего типа. Благодаря использованию пользовательских типов можно делать программы более гибкими: для этого иногда достаточно изменить typedef-инструкции с помощью макросов подстановки (#define). Использование пользовательских типов позволяет также улучшить читабельность кода, поскольку для стандартных типов данных с помощью typedef можно использовать собственные описательные имена. Общий формат записи инструкции для создания пользовательского типа:

```
typedef тип новое_имя;
```

Здесь элемент *тип* означает любой допустимый тип данных, а элемент *новое_имя* - новое имя для этого типа. Важно отметить, что новое имя определяется только в качестве дополнения к существующему имени типа, а не для его замены. В языке MQL5 с помощью typedef можно создавать указатели на функции.

Указатель на функцию

Указатель на функцию в общем виде определятся форматом записи

```
typedef тип_результата_функции (*Имя_типа_функции)(список_типов_входных параметров)
```

где после слова typedef задается сигнатура функции - количество и тип входных параметров, а также тип возвращаемого функцией результата. В качестве объяснения приведем простой пример создания и использования указателя на функцию:

```
//--- объявим указатель на функцию, которая принимает два параметра типа int
   typedef int (*TFunc)(int,int);
//--- TFunc является типом и мы можем объявить переменную-указатель на функцию
   TFunc func_ptr; // указатель на функцию
//--- объявим функции, которые соответствуют описанию TFunc
   int sub(int x,int y) { return(x-y); }  // вычитание одного числа из другого
   int add(int x,int y) { return(x+y); }  // сложение двух чисел
   int neg(int x)       { return(~x);  }  // инвертирование битов в переменной
//--- в переменную func_ptr можно сохранить адрес функции, чтобы в дальнейшем ее вызы
   func_ptr=sub;
   Print(func_ptr(10,5));
   func_ptr=add;
   Print(func_ptr(10,5));
   func_ptr=neg;           // ошибка: neg не имеет тип  int (int,int)
   Print(func_ptr(10));    // ошибка: должно быть два параметра
```

В данном примере переменной *func_ptr* можно присвоить функции *sub* и *add*, поскольку они имеют по два входных параметра типа int, как это указано в определении указателя на функцию *TFunc*. А вот функция *neg* не может быть присвоена указателю func_ptr, так как ее сигнатура отличается.

Организации событийных моделей в пользовательском интерфейсе

© 2000-2025, MetaQuotes Ltd.

С помощью указателей на функции удобно строить обработку событий при создании пользовательского интерфейса. Покажем на примере из раздела CButton, как можно создавать кнопки и добавлять в них функции для обработки нажатия. Сначала определим указатель на функцию *TAction*, которая будет вызываться по нажатию кнопки, и создадим три функции в соответствии с описанием *TAction*.

```
//--- создадим пользовательский тип функции
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Открывает файл                                                  |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("Вызвана функция %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Сохраняет файл                                                  |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("Вызвана функция %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Закрывает файл                                                  |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("Вызвана функция %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
```

Затем произведем класс MyButton от CButton, в котором добавим член *TAction*, являющийся указателем на функцию.

```
//+------------------------------------------------------------------+
//| Создадим свой класс кнопки с функцией обработки событий          |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // обработчик событий графика
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- конструктор с указанием текста кнопки и указателя на функцию для обработки со
                     MyButton(string text, TAction act)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Text(text);
      m_action=act;
     }
   //--- установка собственной функции, которая будет вызываться из обработчика событи
   void              SetAction(TAction act){m_action=act;}
   //--- стандартный обработчик событий графика
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL & lparam==Id())
        {
         //--- вызовем собственный обработчик m_action()
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- вернем  результат вызова обработчика из родительского класса CButton
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
```

Далее создадим производный класс CControlsDialog от CAppDialog, в котором добавим массив *m_buttons* для хранения кнопок типа *MyButton*, а также методы *AddButton(MyButton &button)* и *CreateButtons()*.

```
//+------------------------------------------------------------------+
//| Класс CControlsDialog                                            |
//| Назначение: графическая панель для управления приложением        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // массив кнопок
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- добавление кнопки
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- создание кнопок
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Создание объекта CControlsDialog на графике                      |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
```

© 2000-2025, 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
//+------------------------------------------------------------------+
//| Создание и добавление кнопок на панель CControlsDialog           |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
//--- расчет координат кнопок
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- добавим объекты кнопок вместе с указателями на функции
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",::Save));
   AddButton(new MyButton("Close",Close));
//--- создадим кнопки графически
   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);
        }
      //--- добавим каждую кнопку в контейнер CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- succeed
   return(true);
  }
```

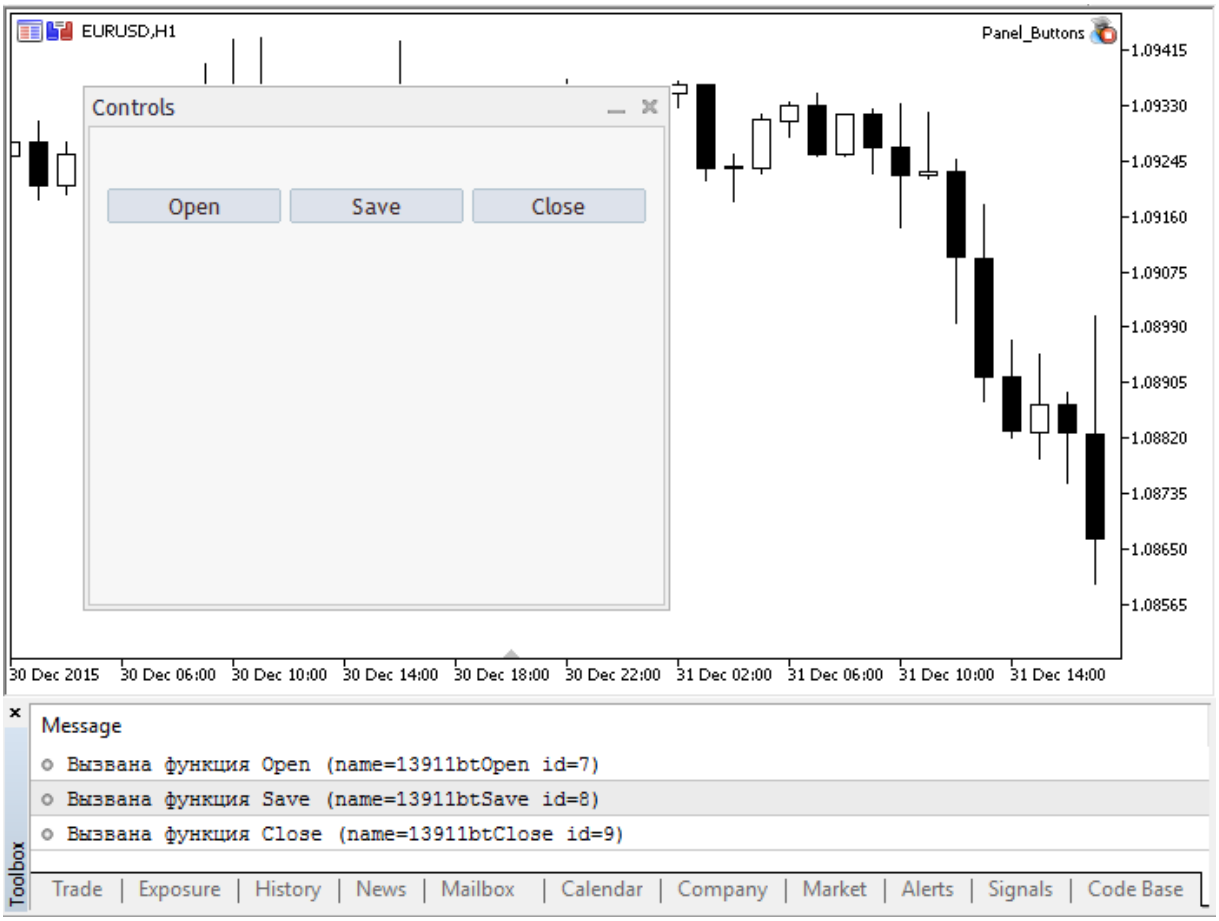
© 2000-2025, MetaQuotes Ltd.

Теперь мы можем написать программу с использованием панели управления CControlsDialog, в которой создается 3 кнопки "Open", "Save" и "Close". При нажатии на кнопку вызывается соответствующая ей функция, которая прописана в виде указателя на функцию *TAction*.

```
//--- объявим объект на глобальном уровне, чтобы создать его автоматически при запуске
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- теперь создадим объект на графике
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- запускаем приложение
   MyDialog.Run();
//--- успешная инициализация приложения
   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
  {
//--- для событий графика вызываем обработчик из родительского класса (CAppDialog в да
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Внешний вид запущенного приложения и результаты нажатия кнопок представлены на картинке.

© 2000-2025, MetaQuotes Ltd.



Полный исходный код программы

```
//+------------------------------------------------------------------+
//|                                                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 "Панель с несколькими кнопками 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
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 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

//--- создадим пользовательский тип функции
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Открывает файл                                                  |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("Вызвана функция %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Сохраняет файл                                                  |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("Вызвана функция %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Закрывает файл                                                  |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("Вызвана функция %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
//+------------------------------------------------------------------+
//| Создадим свой класс кнопки с функцией обработки событий          |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // обработчик событий графика
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- конструктор с указанием текста кнопки и указателя на функцию для обработки со
                     MyButton(string text,TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- установка собственной функции, которая будет вызываться из обработчика событи
```

© 2000-2025, MetaQuotes Ltd.

```
   void              SetAction(TAction act){m_action=act;}
   //--- стандартный обработчик событий графика
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL & lparam==Id())
        {
         //--- вызовем собственный обработчик
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- вернем  результат вызова обработчика из родительского класса CButton
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
//+------------------------------------------------------------------+
//| Класс CControlsDialog                                            |
//| Назначение: графическая панель для управления приложением        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // массив кнопок
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- добавление кнопки
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- создание кнопок
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Создание объекта CControlsDialog на графике                      |
//+------------------------------------------------------------------+
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());
//---
  }
//+------------------------------------------------------------------+
//| Создание и добавление кнопок на панель CControlsDialog           |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- расчет координат кнопок
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- добавим объекты кнопок вместе с указателями на функции
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",::Save));
   AddButton(new MyButton("Close",Close));
//--- создадим кнопки графически
   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);
        }
      //--- добавим каждую кнопку в контейнер CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- succeed
   return(true);
  }
//--- объявим объект на глобальном уровне, чтобы создать его автоматически при запуске
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- теперь создадим объект на графике
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- запускаем приложение
   MyDialog.Run();
//--- успешная инициализация приложения
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy dialog
   MyDialog.Destroy(reason);
```

© 2000-2025, 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
  {
//--- для событий графика вызываем обработчика из родительского класса (CAppDialog в д
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Смотри также

Переменные, Функции

© 2000-2025, MetaQuotes Ltd.

Указатели объектов

В MQL5 существует возможность динамически создавать объекты сложного типа. Это делается при помощи оператора new, который возвращает описатель созданного объекта. Описатель имеет размер 8 байт. Синтаксически описатели объектов в MQL5 похожи на указатели в C++.

Пример:

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

В отличие от C++, переменная hobject из вышеприведенного примера не является указателем на память, а является дескриптором объекта. Кроме того, в языке MQL5 все объекты в параметрах функции обязательно должны передаваться по ссылке. Примеры передачи объектов в качестве параметра функции:

```
class Foo
  {
public:
   string            m_name;
   int               m_id;
   static int        s_counter;
   //--- конструкторы и деструкторы
                     Foo(void){Setup("noname");};
                     Foo(string name){Setup(name);};
                    ~Foo(void){};
   //--- инициализируем объект 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()
  {
//--- объявим объект как переменную с автоматическим созданием
   Foo foo1;
//--- вариант передачи объекта по ссылке
   PrintObject(foo1);

//--- объявим указатель на объект и создадим его с помощью оператора 'new'
   Foo *foo2=new Foo("foo2");
//--- вариант передачи указателя на объект по ссылке
   PrintObject(foo2); // указатель на объект автоматически преобразуется компилятором

//--- объявим массив объектов Foo
```

© 2000-2025, MetaQuotes Ltd.

```
   Foo foo_objects[5];
//--- вариант передачи массива объектов
   PrintObjectsArray(foo_objects); // отдельная функция для передачи массива объектов

//--- объявим массив указателей на объекты типа Foo
   Foo *foo_pointers[5];
   for(int i=0;i<5;i++)
      foo_pointers[i]=new Foo("foo_pointer");
//--- вариант передачи массива указателей
   PrintPointersArray(foo_pointers); // отдельная функция для передачи массива указате

//--- перед завершением работы обязательно удаляем объекты, созданные как указатели
   delete(foo2);
//--- удаляем массив указателей
   int size=ArraySize(foo_pointers);
   for(int i=0;i<5;i++)
      delete(foo_pointers[i]);
//---
  }
//+------------------------------------------------------------------+
//|  Объекты всегда передаются по ссылке                             |
//+------------------------------------------------------------------+
void PrintObject(Foo &object)
  {
   Print(__FUNCTION__,": ",object.m_id," Object name=",object.m_name);
  }
//+------------------------------------------------------------------+
//| Передача массива объектов                                        |
//+------------------------------------------------------------------+
void PrintObjectsArray(Foo &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
//| Передача массива указателей на объект                            |
//+------------------------------------------------------------------+
void PrintPointersArray(Foo* &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
```

Проверка указателя перед использованием

© 2000-2025, MetaQuotes Ltd.

Попытка обращения к некорректному указателю приводит к критическому завершению программы. Для проверки указателя перед использованием служит функция CheckPointer. Указатель может быть некорректным в следующих случаях:

- указатель равен NULL;
- объект был уничтожен при помощи оператора delete.

Данную функцию можно использовать как проверку указателя на корректность. Значение, отличное от нуля, гарантирует, что по этому указателю можно получить доступ к данным.

```
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()
 {
//--- создадим неинициализованный объект
  CMyObject *pointer;
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

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

//--- удалим объект
  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
```

© 2000-2025, MetaQuotes Ltd.

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

Для быстрой проверки указателя можно также использовать оператор "!" (LNOT), который проверяет его на валидность через неявный вызов функции CheckPointer. Это позволяет писать код более кратко и наглядно. Вот как будут выглядеть проверки из предыдущего примера:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- создадим неинициализованный объект
  CMyObject *pointer;
  if(!pointer)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

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

//--- удалим объект
  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
*/
```

Для быстрой проверки на NULL следует использовать оператор "==". Например: ptr==NULL или ptr!=NULL.

Смотри также

Переменные, Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Ссылки. Модификатор & и ключевое слово this

Передача параметров по ссылке

В MQL5 параметры простых типов можно передавать как по значению, так и по ссылке, в то время как параметры сложных типов всегда передаются по ссылке. Для указания компилятору на необходимость передачи параметра по ссылке, перед именем параметра ставится знак амперсанда **&**.

Передача параметра по ссылке означает передачу адреса переменной, поэтому все изменения, произведенные над переданным по ссылке параметром, сразу же отобразятся и в исходной переменной. Используя передачу параметров по ссылке можно организовать возврат одновременно нескольких результатов из функции. Чтобы предотвратить изменение переданного по ссылке параметра, необходимо использовать модификатор const.

Таким образом, если входной параметр функции является массивом, объектом структуры или класса, то в заголовке функции после типа переменной и перед ее именем ставится символ '&'.

Пример

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

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

В вышеприведенном примере объявлен класс CDemoClass, который содержит приватный член - массив m_array[] типа double. Объявлена функция setArray(), в которую по ссылке передается массив array[]. Если заголовок функции написать без указания передачи по ссылке, т.е. убрать знак амперсанда, то при попытке компиляции такого кода будет выдано сообщение об ошибке.

Несмотря на то, что массив передается по ссылке, мы не можем произвести присвоение одного массива другому. Необходимо сделать поэлементное копирование содержимого массива-источника в массив-приемник. Наличие символа & для массивов и структур при передаче в качестве параметра функции является обязательным при описании функции.

Ключевое слово this

© 2000-2025, MetaQuotes Ltd.

Переменная типа класса (объект) может передаваться как по ссылке, так и по указателю. Указатель как и ссылка служит для того чтобы получать доступ к объекту. После объявления указателя объекта необходимо применить к нему оператор new для его создания и инициализации.

Зарезервированное слово **this** предназначено для получения ссылки объекта на самого себя, доступной внутри методов класса или структуры. **this** всегда ссылается на объект, в методе которого используется, а выражение GetPointer(this) даёт указатель объекта, членом которого является функция, в которой осуществлен вызов функции GetPointer(). В MQL5 функции не могут возвращать объекты, но разрешено возвращать указатель объекта.

Таким образом, если необходимо, чтобы функция вернула объект, то мы можем вернуть указатель этого объекта в виде GetPointer(this). Добавим в описание класса CDemoClass функцию getDemoClass(), которая возвращает указатель объекта этого класса.

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

public:
   void              setArray(double &array[]);
   CDemoClass       *getDemoClass();
  };
//+------------------------------------------------------------------+
//| заполнение массива                                               |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array,array);
     }
  }
//+------------------------------------------------------------------+
//| возвращает собственный указатель                                 |
//+------------------------------------------------------------------+
CDemoClass *CDemoClass::getDemoClass(void)
  {
   return(GetPointer(this));
  }
```

Структуры не имеют указателей, к ним нельзя применять операторы *new* и *delete*, и нельзя использовать GetPointer(this).

Смотри также

Указатели объектов, Создание и уничтожение объектов, Область видимости и время жизни переменных

© 2000-2025, MetaQuotes Ltd.

Операции и выражения

Некоторым символам и символьным последовательностям придается особое значение. Это - так называемые символы операций, например:

```
+ - * / %      символы арифметических операций
&& ||          символы логических операций
= += *=        символы операций присваивания
```

Символы операций используются в выражениях и имеют смысл тогда, когда им даны соответствующие операнды. Также особое значение придается знакам препинания. Знаки препинания включают круглые скобки, фигурные скобки, запятую, двоеточие и точку с запятой.

Символы операций, знаки препинания и пробелы служат для того, чтобы отделять элементы языка.

В данном разделе рассматриваются следующие темы:

- Выражения
- Арифметические операции
- Операции присваивания
- Операции отношения
- Логические операции
- Побитовые операции
- Другие операции
- Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Выражения

Выражение состоит из одного или нескольких операндов и символов операций. Может записываться в несколько строк.

Примеры:

```
a++; b = 10;          // несколько выражений расположены на одной строчке
//---  одно выражение разбито на несколько строк
x = (y * z) /
    (w + 2) + 127;
```

Выражение, заканчивающееся точкой с запятой (;), является оператором.

Смотри также

Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Арифметические операции

К арифметическим относятся аддитивные и мультипликативные операции:

```
Сумма величин                          i = j + 2;
Вычитание величин                      i = j - 3;
Изменение знака                        x = - x;
Умножение величин                      z = 3 * x;
Частное от деления                     i = j / 5;
Остаток от деления                     minutes = time % 60;
Добавление 1 к значению переменной     i++;
Добавление 1 к значению переменной     ++i;
Вычитание 1 от значения переменной     k--;
Вычитание 1 от значения переменной     --k;
```

Операция инкремента и декремента применяются только к переменным, к константам не применяются. Префиксные инкремент (++i) и декремент (--k) применяются к переменной непосредственно перед использованием этой переменной в выражении.

Постфиксные инкремент (i++) и декремент (k--) применяются к переменной сразу после использования этой переменной в выражении.

Важное замечание

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

Могут возникнуть вычислительные проблемы при переносе вышеуказанного выражения из одной среды программирования в другую (например, из Borland C++ в MQL5). В общем случае порядок вычислений зависит от реализации компилятора. На практике существуют два способа реализации постдекремента (постинкремента):

1. постдекремент (постинкремент) применяется к переменной после вычисления всего выражения;
2. постдекремент (постинкремента) применяется к переменной сразу по месту операции.

В MQL5 в данный момент реализован первый способ вычисления постдекремента (постинкремента). Но даже обладая этим знанием лучше не экспериментировать с использованием данной тонкости.

Примеры:

```
int a=3;
a++;             // верное выражение
int b=(a++)*3;   // неверное выражение
```

Смотри также

Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Операции присваивания

Значением выражения, в которое входит операция присваивания, является значение левого операнда после присваивания:

```
Присваивание значения x переменной y                    y = x;
```

Следующие операции объединяют арифметические или побитовые операции с операцией присваивания:

```
Увеличение значения переменной y на x                   y += x;
Уменьшение значения переменной y на x                   y -= x;
Умножение значения переменной y на x                    y *= x;
Деление значения переменной y на x                      y /= x;
Остаток от деления значения переменной y на x           y %= x;
Сдвиг двоичного представления y вправо на x бит         y >>= x;
Сдвиг двоичного представления y влево на x бит          y <<= x;
Побитовая операция И двоичных представлений y и x       y &= x;
Побитовая операция ИЛИ двоичных представлений y и x     y |= x;
Побитовая операция исключающее ИЛИ
          двоичных представлений y и x                  y ^= x;
```

Побитовые операции производятся только с целыми числами. При выполнении операции логический сдвиг представления y вправо/влево на x бит используются младшие 5 двоичных разрядов значения x, старшие разряды отбрасываются, то есть сдвиг производится на 0-31 бит.

При выполнении операции %= (значение y по модулю x) знак результата совпадает со знаком делимого.

В выражении оператор присваивания может присутствовать много раз. В этом случае обработка выражения ведется справа налево:

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

Сначала переменной x будет присвоено значение 3, затем переменной y будет присвоено значение переменной x, то есть тоже 3.

Смотри также

Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Операции отношения

Логическое значение ЛОЖЬ представляется целым нулевым значением, а значение ИСТИНА представляется любым ненулевым.

Значением выражений, содержащих операции отношения или логические операции, являются ЛОЖЬ(0) или ИСТИНА(1).

```
Истина, если a равно b                      a == b;
Истина, если a не равно b                   a != b;
Истина, если a меньше b                     a < b;
Истина, если a больше b                     a > b;
Истина, если a меньше или равно b           a <= b;
Истина, если a больше или равно b           a >= b;
```

Нельзя сравнивать два вещественных числа на равенство друг другу. В большинстве случаев два вроде бы одинаковых числа могут оказаться неравными из-за разницы значения в 15-ом знаке после запятой. Для корректного сравнения двух вещественных чисел необходимо сравнивать нормализованную разницу этих чисел с нулевым значением.

Пример:

```
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  %.16f   все таки равны",first,third);
     }
  }
// Результат: 0.3000000000000000  0.2999999999999998   все таки равны
```

Смотри также

Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Логические операции

Логическое отрицание НЕ(!)

Операнд операции логического отрицания НЕ(!) должен иметь арифметический тип. Результат равен ИСТИНА(1), если значение операнда есть ЛОЖЬ(0), и равен ЛОЖЬ(0), если операнд не равен ЛОЖЬ(0).

```
if(!a) Print("не 'a'");
```

Логическая операция ИЛИ (||)

Логическая операция ИЛИ (||) значений x и y. Значением выражения является ИСТИНА(1), если истинно (не нуль) значение x или y. В противном случае - ЛОЖЬ(0).

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

Логическая операция И (&&)

Логическая операция И (&&) значений x и y. Значением выражения является ИСТИНА(1), если значения x и y истинны (не нуль). В противном случае - ЛОЖЬ(0).

Короткая оценка логических операций

К логическим выражениям применяется схема так называемой "короткой оценки", то есть, вычисление выражения прекращается в тот момент, когда можно точно оценить результат выражения.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- первый пример короткой оценки
   if(func_false() && func_true())
      Print("Операция &&: Это сообщение вы никогда не увидите");
   else
      Print("Операция &&: Результат первого выражения false, поэтому второе выражение
//--- второй пример короткой оценки
   if(!func_false() || !func_true())
      Print("Операция ||: Результат первого выражения true, поэтому второе выражение в
   else
      Print("Операция ||: Это сообщение вы никогда не увидите");
  }
//+------------------------------------------------------------------+
//| функция всегда возвращает false                                  |
//+------------------------------------------------------------------+
bool func_false()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   Print("Функция func_false()");
   return(false);
  }
//+------------------------------------------------------------------+
//| функция всегда возвращает true                                   |
//+------------------------------------------------------------------+
bool func_true()
  {
   Print("Функция func_true()");
   return(true);
  }
```

Смотри также

Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Побитовые операции

Дополнение до единицы

Дополнение до единицы значения переменной. Значение выражения содержит 1 во всех разрядах, в которых значение переменной содержит 0, и 0 во всех разрядах, в которых значения переменной содержит 1.

```
b = ~n;
```

Пример:

```
   char a='a',b;
   b=~a;
   Print("a = ",a, "  b = ",b);
// Результат будет такой:
// a= 97   b= -98
```

Сдвиг вправо

Двоичное представление x сдвигается вправо на y разрядов. Если сдвигаемое значение имеет беззнаковый тип, то осуществляется логический сдвиг вправо, то есть, освобождающиеся слева разряды будут заполняться нулями.

Если же сдвигаемое значение имеет знаковый тип, то осуществляется арифметический сдвиг вправо, то есть освобождающиеся слева разряды будут заполняться значением знакового бита (если число положительное, то значение знакового бита равно 0, если число отрицательное, то значение знакового бита равно 1)

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

Пример:

```
   char a='a',b='b';
   Print("Before:  a = ",a, "  b = ",b);
//--- произведем сдвиг вправо
   b=a>>1;
   Print("After:   a = ",a, "  b = ",b);
// Результат будет такой:
// Before:  a = 97   b = 98
// After:   a = 97   b = 48
```

Сдвиг влево

Двоичное представление x сдвигается влево на y разрядов; освобождающиеся справа разряды заполняются нулями.

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

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
//--- произведем сдвиг влево
   b=a<<1;
   Print("After:   a = ",a, "  b = ",b);
// Результат будет такой:
// Before:  a = 97   b = 98
// After:   a = 97   b = -62
```

Не рекомендуется производить сдвиг на большее или равное число битов, чем разрядность сдвигаемой переменной, так как результат такой операции не определен.

При выполнении операции сдвига над 32-х битными типами для вычисления сдвига учитываются только младшие 5 бит, а для операций сдвига над 64-х битными типами значения сдвига определяется только младшими 6 битами. Покажем это на примере:

```
   int a=305;     // тип int имеет размер 4 байта и является 32-х битными число
   int b=a;       // сделаем копию
   int shift=37; // зададим переменную сдвига
   shift++;       // увеличим значения на 1, shift=38
   //--- задаем сдвиг на 38 разрядов, но на самом деле будет произведен сдвиг на 6 ра
   a=a>>shift;    // 38 в двоичном представлении будет выглядеть как '100110', младшие
   b=b>>6;        // над переменной b произведен сдвиг на 6 разрядов
   //--- убедимся, что переменные a и b содержат одинаковое значение после операции с
   Print("a = ", a, "    b = ", b);
```

Побитовая операция И

Побитовая операция И двоичных представлений x и y. Значение выражения содержит 1 (ИСТИНА) во всех разрядах, в которых и x, и y содержат не ноль; и 0 (ЛОЖЬ) во всех остальных разрядах.

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

Пример:

```
   char a='a',b='b';
//--- операция И
   char c=a&b;
   Print("a = ",a,"  b = ",b);
   Print("a & b = ",c);
// Результат будет такой:
// a = 97   b = 98
// a & b = 96
```

Побитовая операция ИЛИ

Побитовая операция ИЛИ двоичных представлений x и y. Значение выражения содержит 1 во всех разрядах, в которых x или y не содержит 0, и 0 - во всех остальных разрядах.

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

Пример:

```
   char a='a',b='b';
//--- операция ИЛИ
```

© 2000-2025, MetaQuotes Ltd.

```
   char c=a|b;
   Print("a = ",a,"  b = ",b);
   Print("a | b = ",c);
// Результат будет такой:
// a = 97   b = 98
// a | b = 99
```

Побитовая операция исключающее ИЛИ

Побитовая операция исключающее ИЛИ (eXclusive OR) двоичных представлений x и y. Значение выражения содержит 1 в тех разрядах, в которых x и y имеют разные двоичные значения, и 0 - во всех остальных разрядах.

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

Пример:

```
   char a='a', b='b';
//--- операция исключающее ИЛИ
   char c=a^b;
   Print("a = ",a,"  b = ",b);
   Print("a ^ b = ",c);
// Результат будет такой:
// a = 97   b = 98
// a ^ b = 3
```

Побитовые операции выполняются только с целыми числами.

Смотри также

Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Другие операции

Индексирование ([])

При обращении к i-му элементу массива значением выражения является значение переменной с порядковым номером i.

Пример:

```
array[i] = 3; // Присвоить значение 3 i-му элементу массива array.
```

Индексом массива может быть только целое число. Допускаются не более чем четырехмерные массивы. Индексация каждого измерения производится от 0 до **размер измерения**-1. В частном случае одномерного массива из 50 элементов обращение к первому элементу будет выглядеть как array[0], к последнему элементу - array[49].

При доступе за пределы массива исполняющая подсистема сгенерирует критическую ошибку, и выполнение программы будет остановлено.

Вызов функции с аргументами x1, x2,..., xn

Каждый аргумент может представлять собой константу, переменную или выражение соответствующего типа. Передаваемые аргументы разделяются запятыми и должны находиться внутри круглых скобок, открывающая круглая скобка должна следовать за именем вызываемой функции.

Значением выражения является значение, возвращаемое функцией. Если тип возвращаемого значения функции есть void, то вызов такой функции нельзя помещать справа в операции присвоения. Обратите внимание, что порядок выполнения выражений x1,..., xn гарантируется.

Пример:

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

Операция запятая (,)

Выражения, разделенные запятыми, вычисляются слева направо. Все побочные эффекты вычисления левого выражения могут возникать до вычисления правого выражения. Тип и значение результата совпадают с типом и значением правого выражения. В качестве примера можно рассматривать список передаваемых параметров (см.выше).

© 2000-2025, MetaQuotes Ltd.

Пример:

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

Операция точка (.)

Для прямого доступа к публичным членам структур и классов используется операция точка. Синтаксис

```
Имя_переменной_типа_структуры.Имя_члена
```

Пример:

```
   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;
```

Операция разрешения контекста (::)

Каждая функция в mql5-программе имеет свой контекст исполнения. Например, системная функция Print() выполняется в глобальном контексте. Импортируемые функции вызываются в контексте соответствующего импорта. Функции-методы классов имеют контекст соответствующего класса. Синтаксис операции разрешения контекста:

```
[Имя_контекста]::Имя_функции(параметры)
```

Если имя контекста отсутствует, то это явное указание на использование глобального контекста. В случае отсутствия операции разрешения контекста, функция ищется в ближайшем контексте. В случае отсутствия функции в локальном контексте, поиск производится в глобальном контексте.

Также операция разрешения контекста используется для определения функции-члена класса.

```
тип Имя_класса::Имя_функции(описание_параметров)
   {
// тело функции
   }
```

Если в программе используются или могут в будущем использоваться несколько одноименных функций из разных контекстов исполнения, то может возникнуть неоднозначность. Порядок приоритета вызова функций без явного указания контекста:

© 2000-2025, MetaQuotes Ltd.

1. Методы класса. Если функция с заданным именем в классе не задана, то ищем на следующем уровне.
2. Функции MQL5. Если такой функции в языке нет, то ищем на следующем уровне.
3. Глобальные функции, определенные пользователем. Если такой функции нет, то ищем на следующем уровне.
4. Импортируемые функции. Если функция не найдена среди импортируемых, то компилятор выдает ошибку.

Для устранения неоднозначности вызова функций явно указывайте область видимости с помощью операции разрешения контекста.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
   CCheckContext2 test;
   test.Print();
  }
//+------------------------------------------------------------------+
```

Операция взятия размера типа данных или размера объекта любого типа данных (sizeof)

С помощью операции sizeof можно определить размер памяти которая соответствует идентификатору или типу. Операция sizeof имеет следующий формат:

Пример:

```
sizeof(выражение)
```

В качестве выражения может быть использован любой идентификатор, либо имя типа, заключенное в скобки. Отметим, что не может быть использовано имя типа void, а идентификатор не может относится к полю битов или быть именем функции.

Если в качестве выражения указано имя статического массива (то есть, задана первая размерность), то результатом является размер всего массива (т.е. произведение числа элементов на длину типа). Если в качестве выражения указано имя динамического массива (первая размерность не задана), то результатом будет размер объекта динамического массива.

Когда sizeof применяются к имени типа структуры или класса или к идентификатору имеющему тип структуры или класса, то результатом является фактический размер структуры или класса.

Пример:

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

Вычисления размера происходит на этапе компиляции.

Смотри также

Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Приоритеты и порядок операций

Для каждой группы операций в таблице приоритет одинаков. Чем выше приоритет группы операций, тем выше она расположена в таблице. Порядок выполнения определяет группировку операций и операндов.

Внимание: Приоритет выполнения операций в языке MQL5 соответствует приоритету, принятому в языке C++, и отличается от приоритета, заданного в языке MQL4.

Операция	Описание	Порядок выполнения
() [] .	Вызов функции Выделение элемента массива Выделение элемента структуры	Слева направо
! ~ - ++ -- (тип) sizeof	Логическое отрицание Побитовое отрицание (complement) Изменение знака Увеличение на единицу (increment) Уменьшение на единицу (decrement) Преобразование типа Определение размера в байтах	Справа налево
* / %	Умножение Деление Деление по модулю	Слева направо
+ -	Сложение Вычитание	Слева направо
<< >>	Сдвиг влево Сдвиг вправо	Слева направо
< <= > >=	Меньше, чем Меньше или равно Больше, чем Больше или равно	Слева направо
== !=	Равно Не равно	Слева направо
&	Побитовая операция И	Слева направо
^	Побитовая операция исключающее ИЛИ (eXclude OR)	Слева направо
|	Побитовая операция ИЛИ	Слева направо
&&	Логическая операция И	Слева направо
||	Логическая операция ИЛИ	Слева направо
?:	Условная операция	Справа налево
= *=	Присваивание	Справа налево

© 2000-2025, MetaQuotes Ltd.

Операция	Описание	Порядок выполнения
/= %= += -= <<= >>= &= ^= |=	Умножение с присваиванием Деление с присваиванием Деление по модулю с присваиванием Сложение с присваиванием Вычитание с присваиванием Сдвиг влево с присваиванием Сдвиг вправо с присваиванием Побитовое И с присваиванием Исключающее ИЛИ с присваиванием Побитовое ИЛИ с присваиванием	
,	Запятая	Слева направо

Для изменения порядка выполнения операций применяются круглые скобки, которые имеют высший приоритет.

© 2000-2025, MetaQuotes Ltd.

Операторы

Операторы языка описывают некоторые алгоритмические действия, которые необходимо выполнить для решения задачи. Тело программы — это последовательность таких операторов. Идущие друг за другом операторы разделяются точкой с запятой.

Оператор	Описание
Составной оператор {}	Один или более операторов любого типа, заключенные в фигурные скобки { }
Оператор-выражение (;)	Любое выражение, заканчивающееся точкой с запятой (;)
Оператор return	Прекращает выполнение текущей функции и возвращает управление вызвавшей программе
Условный оператор if-else	Используется при необходимости сделать выбор
Условный оператор ?:	Более простой аналог условного оператора if-else
Оператор выбора switch	Передает управление оператору, который соответствует значению выражения
Оператор цикла while	Выполняет оператор до тех пор, пока проверяемое выражение не станет ложным. Проверка выражения производится перед каждой итерацией
Оператор цикла for	Выполняет оператор до тех пор, пока проверяемое выражение не станет ложным. Проверка выражения производится перед каждой итерацией
Оператор цикла do-while	Выполняет оператор до тех пор, пока проверяемое выражение не станет ложным. Проверка условия окончания производится в конце, после каждого прохода цикла. Тело цикла всегда выполняется по крайней мере один раз.
Оператор break	Прекращает выполнение ближайшего вложенного внешнего оператора switch, while, do-while или for
Оператор continue	Передает управление в начало ближайшего внешнего оператора цикла while, do-while или for
Оператор @	Матричное умножение по правилам линейной алгебры. Он позволяет выполнять произведение матриц, векторов и скалярное произведение векторов.
Оператор new	Создает объект соответствующего размера и возвращает описатель созданного объекта.
Оператор delete	Удаляет созданный оператором new объект

Один оператор может занимать одну или более строк. Два или большее количество операторов могут быть расположены на одной строке. Операторы, управляющие порядком выполнения (if, if-else, switch, while и for), могут быть вложены друг в друга.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
if(Month() == 12)
  if(Day() == 31)
     Print("Happy New Year!");
```

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Составной оператор

Составной оператор (блок) состоит из одного или большего числа операторов любого типа, заключенных в фигурные скобки { }. После закрывающейся фигурной скобки не должно быть точки с запятой (;).

Пример:

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

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор-выражение

Любое выражение, заканчивающееся точкой с запятой (;), является оператором. Далее следуют примеры операторов-выражений.

Оператор присваивания:

Идентификатор = выражение;

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

В выражении оператор присваивания может присутствовать много раз. В этом случае обработка выражения ведется справа налево.

Оператор вызова функции:

Имя_функции (аргумент1,..., аргументN);

```
FileClose(file);
```

Пустой оператор

Состоит только из точки с запятой (;) и используется для обозначения пустого тела управляющего оператора.

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор возврата return

Оператор return прекращает выполнение текущей функции и возвращает управление вызвавшей программе. Результат вычисления выражения возвращается вызываемой функции. Выражение может содержать оператор присваивания.

Пример:

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

В функциях с типом возвращаемого значения void необходимо использовать оператор return без выражения:

```
void SomeFunction()
  {
   Print("Hello!");
   return;     // этот оператор можно удалить
  }
```

Завершающая фигурная скобка функции предполагает неявное исполнение оператора return без выражения.

Можно возвращать простые типы, простые структуры, указатели объектов. При помощи оператора *return* нельзя возвращать любые массивы, объекты классов, переменные типа сложных структур.

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Условный оператор if-else

Оператор IF - ELSE используется при необходимости сделать выбор. Формально синтаксис имеет вид:

```
if (выражение)
     оператор1
else
     оператор2
```

Если выражение истинно, то выполняется оператор1 и управление передается на оператор, следующий за оператором2 (т.е. оператор2 не выполняется). Если выражение ложно, то выполняется оператор2.

Часть else оператора if может опускаться. Поэтому во вложенных операторах if с пропущенной частью else может возникнуть неоднозначность. В этом случае else связывается с ближайшим предыдущим оператором if в том же блоке, не имеющим части else.

Примеры:

```
//--- Часть else относится ко второму оператору if:
if(x>1)
   if(y==2) z=5;
else     z=6;
//--- Часть else относится к первому оператору if
if(x>1)
  {
   if(y==2) z=5;
  }
else        z=6;
//--- Вложенные операторы
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y=4;
  }
else Print("ERROR");
```

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Условный оператор ?:

Общая форма тернарного оператора выглядит так:

```
выражение1 ? выражение2 : выражение3
```

В качестве первого операнда - "выражение1" - может быть использовано любое выражение, результатом которого является значение типа bool. Если результат равен true, то выполняется оператор, заданный вторым операндом, то есть, "выражение2".

Если же первый операнд равен false, то выполняется третий операнд - "выражение3". Второй и третий операнды, то есть "выражение2" и "выражение3", должны возвращать значения одного типа и не должны иметь тип void. Результатом выполнения условного оператора является результат "выражения2" либо результат "выражение3", в зависимости от результата "выражение1".

```
//--- пронормируем разность между ценами открытия и закрытия на дневной размах
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

Эта запись эквивалентна следующей

```
double true_range;
if(High==Low)
   true_range=0;                // если High и Low равны
else
   true_range=(Close-Open)/(High-Low); // если размах ненулевой
```

Ограничения по использованию оператора

Оператор на основании значения "выражение1" должен вернуть одно из двух значений - либо "выражение2", либо "выражение3". Существует ряд ограничений на эти выражения:

1. Нельзя смешивать пользовательский тип с простым типом или перечислением. Для указателя допустимо использовать NULL.
2. Если типы значений простые, то типом оператора будет максимальный тип (смотри Приведение типов).
3. Если одно из значений имеет тип перечисление, а второе является числовым типом, то перечисление заменяется на int и действует второе правило.
4. Если оба значения являются значениями перечислений, то их типы должны быть одинаковыми, и типом оператора будет перечисление.

Ограничения для пользовательских типов (классов или структур):

a) типы должны быть одинаковыми или один должен наследоваться от другого.

b) если типы не одинаковы (наследование), то потомок неявно приводится к родителю, то есть типом оператора будет тип родителя.

c) нельзя смешивать объект и указатель - либо оба выражения являются объектами, либо указателями. Для указателя допустимо использовать NULL.

Примечание

© 2000-2025, MetaQuotes Ltd.

Будьте внимательны при использовании условного оператора в качестве аргумента перегруженной функции, так как тип результата условного оператора определяется на момент компиляции программы. И этот тип определяется как больший тип из типов "выражение2" и "выражение3".

Пример:

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

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

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

   func(Expression1?Expression2:Expression3);    // получим предупреждение компилятора
   func(!Expression1?Expression2:Expression3);   // получим предупреждение компилятора
  }

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

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор-переключатель switch

Сравнивает значение выражения с константами во всех вариантах *case* и передает управление оператору, который соответствует значению выражения. Каждый вариант *case* может быть помечен целой константой, символьной константой или константным выражением. Константное выражение не может включать переменные или вызовы функций. Выражение оператора *switch* должно быть целого типа int или uint.

```
switch(выражение)
  {
   case константа: операторы
   case константа: операторы
      ...
   default: операторы
  }
```

Операторы, связанные с меткой *default*, выполняются, если ни одна из констант в операторах *case* не равна значению выражения. Вариант *default* не обязательно должен быть объявлен и не обязательно должен быть последним. Если ни одна константа не соответствует значению выражения и вариант *default* отсутствует, то не выполняется никаких действий.

Ключевое слово *case* вместе с константой служат просто метками, и если будут выполняться операторы для некоторого варианта *case*, то далее будут выполняться операторы всех последующих вариантов до тех пор, пока не встретится оператор *break*, что позволяет связывать одну последовательность операторов с несколькими вариантами.

Константное выражение вычисляется в период компиляции. Никакие две константы в одном операторе-переключателе не могут иметь одинаковые значения.

Примеры:

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

//---  второй пример
   string res="";
   int i=0;
   switch(i)
     {
```

© 2000-2025, MetaQuotes Ltd.

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

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор цикла while

Оператор while состоит из проверяемого выражения и оператора, который должен быть выполнен:

```
while(выражение)
  оператор;
```

Если выражение истинно, то оператор выполняется до тех пор, пока выражение не станет ложным. Если выражение ложно, то управление передается следующему оператору. Значение выражения определяется до выполнения оператора. Следовательно, если выражение ложно с самого начала, то оператор вообще не выполняется.

Примечание

Если в цикле предполагается обрабатывать большое количество итераций, то рекомендуется проверять факт принудительного завершения программы с помощью функции IsStopped().

Пример:

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

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор цикла for

Оператор for состоит из трех выражений и выполняемого оператора:

```
for(выражение1; выражение2; выражение3)
   оператор;
```

Выражение1 описывает инициализацию цикла. Выражение2 - проверка условия завершения цикла. Если оно истинно, то выполняется оператор тела цикла for. Все повторяется, пока выражение2 не станет ложным. Если оно ложно, цикл заканчивается и управление передается следующему оператору. Выражение3 вычисляется после каждой итерации.

Оператор for эквивалентен следующей последовательности операторов:

```
выражение1;
while(выражение2)
  {
   оператор;
   выражение3;
  };
```

Любое из трех или все три выражения в операторе for могут отсутствовать, однако разделяющие их точки с запятыми (;) опускать нельзя. Если опущено выражение2, то считается, что оно постоянно истинно. Оператор for(;;) представляет собой бесконечный цикл, эквивалентный оператору while(1). Каждое из выражение1 и выражение3 может состоять из нескольких выражений, объединенных оператором запятая ','.

Примечание

Если в цикле предполагается обрабатывать большое количество итераций, то рекомендуется проверять факт принудительного завершения программы с помощью функции IsStopped().

Примеры:

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

© 2000-2025, MetaQuotes Ltd.

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор цикла do while

Циклы for и while производят проверку окончания в начале, а не в конце цикла. Третий оператор цикла do - while проверяет условие окончания в конце, после каждого прохода цикла. Тело цикла всегда выполняется по крайней мере один раз.

```
do
   оператор;
while(выражение);
```

Сначала выполняется оператор, затем вычисляется выражение. Если оно истинно, то оператор выполняется снова и т.д. Если выражение становится ложным, цикл заканчивается.

Примечание

Если в цикле предполагается обрабатывать большое количество итераций, то рекомендуется проверять факт принудительного завершения программы с помощью функции IsStopped().

Пример:

```
//--- вычисление последовательности чисел Фибоначчи
   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++; // без этого оператора получится бесконечный цикл!!
     }
   while(i<counterFibonacci && !IsStopped());
```

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор завершения break

Оператор break прекращает выполнение ближайшего вложенного внешнего оператора switch, while, do-while или for. Управление передается оператору, следующему за заканчиваемым. Одно из назначений этого оператора - закончить выполнение цикла при присваивании некоторой переменной определенного значения.

Пример:

```
//--- поиск первого нулевого элемента
for(i=0;i<array_size;i++)
  if(array[i]==0)
    break;
```

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор продолжения continue

Оператор continue передает управление в начало ближайшего внешнего оператора цикла while, do-while или for, вызывая начало следующей итерации. Этот оператор по действию противоположен оператору break.

Пример:

```
//--- сумма всех ненулевых элементов
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);
  }
```

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор матричного произведения @

Оператор @ в языке MQL5 реализует матричное умножение по правилам линейной алгебры. Он позволяет выполнять произведение матриц, векторов и скалярное произведение векторов.

Поддерживаемые типы элементов:

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

Важно: типы элементов в левом и правом операндах должны совпадать.

Примеры использования

1. Матричное умножение (матрица × матрица)

```
matrix A(2, 3);
matrix B(3, 2);
matrix C = A @ B; // Результат: матрица C размером [2,2]
```

2. Матричное умножение (матрица × вектор)

```
matrix M(2, 3);
vector V(3);
vector R = M @ V; // Результат: вектор R из 2 элементов
```

3. Матричное умножение (вектор × матрица)

```
matrix M(2, 3);
vector V(1, 2);
vector R = V @ M; // Результат: вектор R из 3 элементов
```

4. Скалярное произведение (вектор × вектор)

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

Примечание

Размерности должны соответствовать правилам умножения: число столбцов в первом операнде = числу строк во втором.

В случае ошибки размерностей будет выброшено исключение на этапе выполнения.

Векторы слева считаются горизонтальными (1×n).

© 2000-2025, MetaQuotes Ltd.

Векторы справа считаются вертикальными (n×1).

Приоритет опреации @ соответствует приоритету умножения, т.е. для записи вида D=C+A@B, сначала будет произведено матричное умножение T=A@B, затем поэлементное сложение D=C+T.

Смотри также

MatMul, GeMM, Kron, Dot,

© 2000-2025, MetaQuotes Ltd.

Оператор создания объекта new

Оператор new автоматически создает объект соответствующего размера, вызывает конструктор объекта и возвращает описатель созданного объекта. В случае неудачи оператор возвращает нулевой описатель, который можно сравнивать с константой NULL.

Оператор *new* может быть применен только к объектам класса, к структурам он не применим.

Оператор не применяется для создания массивов объектов. Для этого следует использовать функцию ArrayResize().

Пример:

```
//+------------------------------------------------------------------+
//| Создание фигуры                                                  |
//+------------------------------------------------------------------+
void CTetrisField::NewShape()
  {
   m_ypos=HORZ_BORDER;
//--- случайным образом создаём одну из 7 возможных фигур
   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;
     }
//--- отрисовываем
   if(m_shape!=NULL)
     {
      //--- начальные установки
      m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
      m_shape.SetYPos(m_ypos);
      m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
      //--- отрисуем
      m_shape.Draw();
     }
//---
  }
```

Следует отметить, что описатель объекта не является указателем на память.

Объект, созданный с помощью оператора new, должен быть явно уничтожен оператором delete.

Смотри также

© 2000-2025, MetaQuotes Ltd.

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Оператор уничтожения объекта delete

Оператор delete удаляет созданный оператором new объект, вызывает деструктор соответствующего класса и освобождает память, занимаемую объектом. В качестве операнда используется действительный описатель существующего объекта. После выполнения операции delete описатель объекта становится недействительным.

Пример:

```
//--- удаляем уложенную фигуру
delete m_shape;
m_shape=NULL;
//--- создаём новую фигуру
NewShape();
```

Смотри также

Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Функции

Всякая задача может быть разбита на подзадачи, каждую из которых можно либо непосредственно представить в виде кода, либо разбить на еще более мелкие подзадачи. Данный метод называется *пошаговым уточнением*. Функции служат для записи программного кода этих непосредственно решаемых подзадач. Код, описывающий, что делает функция, называется *определением функции:*

```
заголовок_функции
  {
   инструкции
  }
```

Все, что находится перед первой фигурной скобкой, составляет *заголовок* определения функции, а то, что находится между фигурными скобками, является *телом* определения функции. Заголовок функции включает в себя описание типа возвращаемого значения, имени (идентификатора) и формальных параметров. Количество параметров, передаваемых в функцию, ограничено и не может превышать 64.

Функция может вызываться из других частей программы столько раз, сколько необходимо. По сути, возвращаемый тип, идентификатор функции и типы параметров составляют *прототип функции*.

Прототип функции - это объявление функции, но не ее определение. Благодаря явному объявлению возвращаемого типа и списка типов аргументов, при обращении к функциям возможны строгая проверка типов и неявные преобразования типов. Особенно часто объявления функций используются в классах для улучшения читаемости кода.

Определение функции должно точно соответствовать ее объявлению. Каждая объявленная функция должна быть определена.

Пример:

```
double                          // тип возвращаемого значения
linfunc (double a, double b) // имя функции и список параметров
  {
                                // составной оператор
   return (a + b);              // возвращаемое значение
  }
```

Оператор return может возвращать значение выражения, стоящего в этом операторе. Значение выражения при необходимости преобразуется к типу результата функции. Можно возвращать простые типы, простые структуры, указатели объектов. При помощи оператора *return* нельзя возвращать любые массивы, объекты классов, переменные типа сложных структур.

Функция, которая не возвращает значения, должна быть описана как имеющая тип void.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
  }
```

Параметры, передаваемые в функцию, могут иметь умолчательные значения, которые задаются константами соответствующего типа.

Пример:

```
int somefunc(double a,
             double d=0.0001,
             int n=5,
             bool b=true,
             string s="passed string")
  {
   Print("Обязательный параметр a= ",a);
   Print("Переданы следующие параметры: d = ",d," n = ",n," b = ",b," s = ",s);
   return(0);
  }
```

Если какому-либо параметру было назначено умолчательное значение, то все последующие параметры также должны иметь умолчательное значение.

Пример неправильного объявления:

```
int somefunc(double a,
             double d=0.0001,     // объявлено значение по умолчанию 0.0001
             int n,               // значение по умолчанию не указано !
             bool b,              // значение по умолчанию не указано !
             string s="passed string")
  {
  }
```

Смотри также

Перегрузка, Виртуальные функции, Полиморфизм

© 2000-2025, MetaQuotes Ltd.

Вызов функции

Если некоторое имя, которое не было описано ранее, появляется в выражении и за ним следует левая круглая скобка, то оно по контексту считается именем некоторой функции.

```
имя_функции (x1, x2,..., xn)
```

Аргументы (формальные параметры) передаются по значению, т. е. каждое выражение x1, . . . , xn вычисляется и значение передается функции. Порядок вычисления выражений и порядок загрузки значений не гарантируются. Во время выполнения производится проверка числа и типа аргументов, переданных функции. Такой способ обращения к функции называется вызовом по значению.

Вызов функции - это выражение, значением которого является значение, возвращаемое функцией. Описанный тип функции должен соответствовать типу возвращаемого значения. Функция может быть объявлена или описана в любом месте программы на глобальном уровне, то есть, вне других функций. Функция не может быть объявлена или описана внутри другой функции.

Примеры:

```
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);
  }
```

При вызове функции, имеющей умолчательные параметры, список передаваемых параметров можно ограничить не ранее первого умолчательного параметра.

Примеры:

```
void somefunc(double init,
              double sec=0.0001, //определены значения по умолчанию
              int level=10);
//...
somefunc();                       // неправильный вызов. первый обязательный параметр д
somefunc(3.14);                   // правильный вызов
somefunc(3.14,0.0002);            // правильный вызов
somefunc(3.14,0.0002,10);         // правильный вызов
```

При вызове функции нельзя пропускать параметры, даже имеющие умолчательные значения:

```
somefunc(3.14, , 10);           // неправильный вызов -> второй параметр пропущен.
```

Смотри также

Перегрузка, Виртуальные функции, Полиморфизм

© 2000-2025, MetaQuotes Ltd.

Передача параметров

Существует два метода, с помощью которых машинный язык может передавать аргумент подпрограмме (функции). Первый способ - передача параметра по значению. Этот метод копирует значение аргумента в формальный параметр функции. Поэтому любые изменения этого параметра внутри функции не имеют никакого влияния на соответствующий аргумент вызова.

```
//+------------------------------------------------------------------+
//| передача параметров по значению                                  |
//+------------------------------------------------------------------+
double FirstMethod(int i,int j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a и b перед вызовом:",a," ",b);
   double d=FirstMethod(a,b);
   Print("a и b после вызова:",a," ",b);
  }
//--- результат выполнения скрипта
//  a и b перед вызовом: 14 8
//  a и b после вызова: 14 8
```

Второй способ - передача аргумента по ссылке. В этом случае ссылка на параметр (а не его значение) передается параметру функции. Внутри функции она используется для того, чтобы обратиться к фактическому параметру, указанному в вызове. Это означает, что изменения параметра будут влиять на аргумент, использованный для вызова функции.

```
//+------------------------------------------------------------------+
//| передача параметров по ссылке                                    |
//+------------------------------------------------------------------+
double SecondMethod(int &i,int &j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a и b перед вызовом:",a," ",b);
   double d=SecondMethod(a,b);
   Print("a и b после вызова:",a," ",b);
  }
//+------------------------------------------------------------------+
//--- результат выполнения скрипта
//  a и b перед вызовом: 14 8
//  a и b после вызова: 28 4
```

MQL5 использует оба метода, за одним исключением: массивы, переменные типа структур и объекты классов всегда передаются по ссылке. Для того чтобы исключить изменения фактических параметров (аргументов, переданных при вызове функции) необходимо использовать спецификатор доступа const. При попытке изменить содержимое переменной, объявленной со спецификатором *const*, компилятор выдаст ошибку.

Примечание

Необходимо помнить, что параметры передаются в функцию задом наперед, то есть сначала вычисляется и передается самый последний параметр, затем предпоследний и так далее. Последним по очереди вычисляется и передается параметр, стоящий первым после открывающей скобки.

Пример:

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

   func(a[i],a[i++],"Первый вызов (i = "+string(i)+")");
   func(a[i++],a[i],"Второй вызов (i = "+string(i)+")");
// Результат:
// Первый вызов(i=0) : par1 = 1     par2 = 0
// Второй вызов(i=1) : par1 = 1     par2 = 1

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

© 2000-2025, MetaQuotes Ltd.

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

При первом вызове в приведенном примере сначала переменная ***i*** участвует в конкатенации строк:

```
"Первый вызов (i = "+string(i)+")"
```

При этом ее значение не меняется. Затем переменная **i** участвует в вычислении элемента массива ***a[i++]***, то есть после взятия i-го элемента массива переменная ***i*** инкрементируется. И только после этого вычисляется первый параметр с измененным значением переменной ***i***.

Во втором вызове при вычислении всех трех параметров используется одно и тоже значение i, измененное на первом вызове функции, и только после вычисления первого параметра переменная ***i*** опять изменяется.

Смотри также

Область видимости и время жизни переменных, Перегрузка, Виртуальные функции, Полиморфизм

© 2000-2025, MetaQuotes Ltd.

Перегрузка функций

Обычно в названии функции стремятся отобразить ее основное назначение. Читабельные программы, как правило, содержат разнообразные и грамотно подобранные идентификаторы. Иногда различные функции используются для одних и тех же целей. Например, рассмотрим функцию, которая вычисляет среднее значение массива чисел двойной точности, и такую же функцию, но оперирующую массивом целых чисел. И ту, и другую удобно назвать AverageFromArray:

```
//+------------------------------------------------------------------+
//| вычисление среднего для массива типа 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];    // сложение для double
     }
   aver=sum/size;       // просто делим сумму на количество
//---
   Print("Вычисление среднего для массива типа double");
   return aver;
  }
//+------------------------------------------------------------------+
//| вычисление среднего для массива типа 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];     // сложение для int
   aver=(double)sum/size;// приведем сумму к типу double и разделим
//---
   Print("Вычисление среднего для массива типа int");
   return aver;
  }
```

Каждая функция содержит вывод сообщения посредством функции Print();

```
   Print("Вычисление среднего для массива типа int");
```

© 2000-2025, MetaQuotes Ltd.

Компилятор выбирает нужную функцию в соответствии с типами аргументов и их количеством. Правило, по которому осуществляется этот выбор, называется *алгоритмом соответствия сигнатуре*. Под сигнатурой понимается список типов, который используется в объявлении функции.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
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);
  }
//--- Результат работы скрипта
// Вычисление среднего для массива типа int
// Вычисление среднего для массива типа double
// int_aver= 3.00000000    double_aver= 3.30000000
```

Перегрузка функций - это создание нескольких функций с одним именем, но с разными параметрами. Это означает, что в перегружаемых вариантах функции разным должно быть количество аргументов и/или их тип. Выбор конкретного варианта функции зависит от типов аргументов, полученных функцией. Конкретная функция выбирается в зависимости от соответствия списка аргументов при вызове функции списку параметров в объявлении функции.

Когда вызывается перегруженная функция, компилятор должен иметь алгоритм для выбора надлежащей функции. Алгоритм, который выполняет этот выбор, зависит от того, преобразования какого типа присутствуют. Наилучшее соответствие должно быть уникальным. Найденная функция должна быть наилучшим выбором среди остальных вариантов хотя бы по одному аргументу, и, в то же время, по остальным аргументам она должна подходить не хуже остальных.

Ниже приведен алгоритм соответствия для каждого аргумента.

Алгоритм выбора перегруженной функции

1. Использовать строгое соответствие (если это возможно).
2. Попробовать стандартное повышение типа.
3. Попробовать стандартное преобразование типа.

Стандартное повышение типа лучше, чем остальные стандартные преобразования. Повышение - это преобразование float в double, а также bool, char, short или enum в int. Кроме того, к стандартным преобразованиям относятся преобразования массивов похожих целых типов. Похожими типами являются: bool, char, uchar, так как все три типа являются однобайтовыми целыми; двубайтовые целые short и ushort; 4-байтовые целые int, uint и color; long, ulong и datetime.

© 2000-2025, MetaQuotes Ltd.

Несомненно, строгое соответствие является наилучшим. Для достижения такого соответствия могут использоваться приведения. Компилятор не справится с двусмысленной ситуацией. Поэтому не следует полагаться на тонкие различия в типах и неявные преобразования, которые делают перегруженную функцию неясной.

Если вы сомневаетесь, используйте явные преобразования для обеспечения строгого соответствия.

Примеры перегруженных функций в MQL5 вы можете увидеть на примере функций ArrayInitialize().

Правила перегрузки функций применимы к перегрузке методов классов.

Перегрузка системных функций допускается, но при этом следует следить за тем, чтобы компилятор точно мог выбрать нужную функцию. Для примера, мы можем перегрузить системную функцию MathMax() 4-мя различными способами, но только два варианта будут корректными.

Пример:

```
// 1. перегрузка допустима - функция отличается от встроенной в язык MathMax() по коли
double MathMax(double a,double b,double c);

// 2. перегрузка недопустима!
// количество параметров разное, но последний имеет значение по умолчанию
// это приводит к сокрытию системной функции при вызове, что недопустимо
double MathMax(double a,double b,double c=DBL_MIN);

// 3. перегрузка допустима - нормальная перегрузка по типу параметров a и b
int MathMax(int a,int b);

// 4. перегрузка недопустима!
// количество и тип параметров не отличается от исходной double MathMax(double a,doubl
int MathMax(double a,double b);
```

Смотри также

Перегрузка, Виртуальные функции, Полиморфизм

© 2000-2025, MetaQuotes Ltd.

Перегрузка операций

Для удобства чтения и написания кода разрешается перегрузка некоторых операций. Оператор перегрузки записывается с помощью ключевого слова operator. Разрешена перегрузка следующих операций:

- бинарные +,-,/,*,%,<<,>>,==,!=,<,>,<=,>=,=,+=,-=,/=,*=,%=,&=,|=,^=,<<=,>>=,&&,||,&,|,^;
- унарные +,-,++,--,!,~;
- оператор присваивания =;
- оператор индексации [].

Перегрузка операций позволяет использовать операционную нотацию (запись в виде простых выражений) к сложным объектам - структурам и классам. Запись выражений с использованием перегруженных операций упрощает восприятие исходного кода, так как более сложная реализация сокрыта.

Для примера рассмотрим широко применяемые в математике комплексные числа, которые состоят из действительной и мнимой части. В языке MQL5 нет типа данных для представления комплексных чисел, но есть возможность создать новый тип данных в виде структуры или класса. Объявим структуру complex и определим в ней четыре метода, реализующие четыре арифметические операции:

```
//+------------------------------------------------------------------+
//| Структура для операций с комплексными числами                    |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // действительная часть
   double            im; // мнимая часть
   //--- конструкторы
                     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) { }
   //--- арифметические операции
   complex           Add(const complex &l,const complex &r) const;  // сложение
   complex           Sub(const complex &l,const complex &r) const;  // вычитание
   complex           Mul(const complex &l,const complex &r) const;  // умножение
   complex           Div(const complex &l,const complex &r) const;  // деление
  };
```

Теперь мы можем объявлять в своем коде переменные, представляющие комплексные числа, и работать с ними.

Например так:

```
void OnStart()
  {
//--- объявим и инициализируем переменные комплексного типа
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- сложим два числа
   complex z;
   z=a.Add(a,b);
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- умножим два числа
   z=a.Mul(a,b);
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- разделим два числа
   z=a.Div(a,b);
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
```

Но было бы удобнее для привычных арифметических операций с комплексными числами использовать привычные операторы "+","-","*" и "/".

Ключевое слово operator используется для того чтобы определить функцию-член, осуществляющую преобразование типа. Унарные и бинарные операции для переменных-объектов класса могут быть перегружены как нестатические функции-члены. Они неявно действуют на объект класса.

Большинство бинарных операций можно перегружать как обычные функции, принимающие один или оба аргумента в виде переменной класса или в виде указателя на объект данного класса. Для нашего типа complex перегрузка в объявлении будет выглядеть так:

```
   //--- операторы
   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)); }
```

Полный пример скрипта:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим и инициализируем переменные комплексного типа
   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;
//--- сложим два числа
   complex z=a+b;
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- умножим два числа

   z=a*b;
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- разделим два числа
```

© 2000-2025, MetaQuotes Ltd.

```
   z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
//+------------------------------------------------------------------+
//| Структура для операций с комплексными числами                    |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // действительная часть
   double            im; // мнимая часть
   //--- конструкторы
                     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) { }
   //--- арифметические операции
   complex           Add(const complex &l,const complex &r) const;  // сложение
   complex           Sub(const complex &l,const complex &r) const;  // вычитание
   complex           Mul(const complex &l,const complex &r) const;  // умножение
   complex           Div(const complex &l,const complex &r) const;  // деление
   //--- бинарные операторы
   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)); }
  };
//+------------------------------------------------------------------+
//| Сложение                                                         |
//+------------------------------------------------------------------+
complex complex::Add(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re+r.re;
   res.im=l.im+r.im;
//--- результат
   return res;
  }
//+------------------------------------------------------------------+
//| Вычитание                                                        |
//+------------------------------------------------------------------+
complex complex::Sub(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re-r.re;
   res.im=l.im-r.im;
//--- результат
```

© 2000-2025, MetaQuotes Ltd.

```
   return res;
  }
//+------------------------------------------------------------------+
//| Умножение                                                        |
//+------------------------------------------------------------------+
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;
//--- результат
   return res;
  }
//+------------------------------------------------------------------+
//| Деление                                                          |
//+------------------------------------------------------------------+
complex complex::Div(const complex &l,const complex &r) const
  {
//--- пустое комплексное число
   complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- проверка на ноль
   if(r.re==0 && r.im==0)
     {
      Print(__FUNCTION__+": number is zero");
      return(res);
     }
//--- вспомогательные переменные
   double e;
   double f;
//--- выбор варианта вычисления
   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;
     }
//--- результат
   return res;
  }
```

© 2000-2025, MetaQuotes Ltd.

Большинство унарных операций для классов можно перегружать как обычные функции, принимающие единственный аргумент-объект класса или указатель на него. Добавим перегрузку унарных операций "-" и "!".

```
//+------------------------------------------------------------------+
//| Структура для операций с комплексными числами                    |
//+------------------------------------------------------------------+
struct complex
  {
   double            re;       // действительная часть
   double            im;       // мнимая часть
...
   //--- унарные операторы
   complex operator-()  const; // унарный минус
   bool    operator!()  const; // отрицание
  };
...
//+------------------------------------------------------------------+
//| Перегрузка оператора "унарный минус"                             |
//+------------------------------------------------------------------+
complex complex::operator-() const
  {
   complex res;
//---
   res.re=-re;
   res.im=-im;
//--- результат
   return res;
  }
//+------------------------------------------------------------------+
//| Перегрузка оператора "логическое отрицание"                      |
//+------------------------------------------------------------------+
bool complex::operator!() const
  {
//--- действительная и мнимая часть комплексного числа равны нулю?
   return (re!=0 && im!=0);
  }
```

Теперь мы можем проверять значение комплексного числа на ноль и получать отрицательное значение:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим и инициализируем переменные комплексного типа
```

© 2000-2025, 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);
//--- разделим два числа
   complex z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//--- комплексное число по умолчанию равно нулю (в конструкторе по умолчанию re==0 и i
   complex zero;
   Print("!zero=",!zero);
//--- присвоим отрицательное значение
   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);
//--- еще раз проверим на равенство нулю
   Print("!zero=",!zero);
//---
  }
```

Обратите внимание, что нам не пришлось в данном случае перегружать операцию присваивания "=", так как структуры простых типов можно копировать друг в друга напрямую. Таким образом, теперь мы можем писать код для расчетов с участием комплексных чисел в привычной манере.

Перегрузка оператора индексирования позволяет получать значения массивов, заключенных в объект, более простым и привычным способом, и это также способствует лучшей читаемости и пониманию исходного кода программ. Например, нам необходимо обеспечить доступ к символу в строке по указанной позиции. Строка в языке MQL5 является отдельным типом string, который не является массивом символов, но с помощью перегруженной операции индексации в созданном классе CString мы можем обеспечить простую и прозрачную работу:

```
//+------------------------------------------------------------------+
//| Класс для доступа к символам в строке как в массиве символов     |
//+------------------------------------------------------------------+
class CString
  {
   string            m_string;

public:
                     CString(string str=NULL):m_string(str) { }
   ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- массив для получения символов из строки
   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;
//--- составим фразу,набрав символы из переменной str
```

© 2000-2025, MetaQuotes Ltd.

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

Другой пример перегрузки операции индексирования - работа с матрицами. Матрица представляет собою двумерный динамический массив, размеры массивов заранее неопределены. Поэтому нельзя объявить массив вида array[][] без указания размера второго измерения и затем передавать этот массив в качестве параметра. Выходом может служить специальный класс CMatrix, который содержит в себе массив объектов класса CRow.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- операции сложения и умножения матриц
   CMatrix A(3),B(3),C();
//--- готовим массивы под строки
   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};
//--- заполняем матрицы
   A[0]=a1; A[1]=a2; A[2]=a3;
   B[0]=b1; B[1]=b2; B[2]=b3;
//--- выведем матрицы в журнал "Эксперты"
   Print("---- элементы матрицы A");
   Print(A.String());
   Print("---- элементы матрицы B");
   Print(B.String());

//--- сложение матриц
   Print("---- сложение матриц A и B");
   C=A+B;
//--- вывод форматированного строкового представления
   Print(C.String());

//--- умножение матриц
   Print("---- произведение матриц A и B");
   C=A*B;
   Print(C.String());

//--- а теперь  покажем как получать значения в стиле динамических массивов matrix[i]
   Print("Выводим значения матрицы C поэлементно");
//--- перебираем в цикле строки матрицы - объекты CRow
   for(int i=0;i<3;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      string com="| ";
      //--- формируем для значения строки из матрицы
      for(int j=0;j<3;j++)
        {
         //--- получим элемент матрицы по номерам строки и столбца
         double element=C[i][j];// [i] - доступ к CRow в массиве m_rows[] ,
                                // [j] - перегруженный оператор индексации в CRow
         com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
        }
      com+="|";
      //--- выводим значения строки
      Print(com);
     }
  }
//+------------------------------------------------------------------+
//| Класс "Строка"                                                   |
//+------------------------------------------------------------------+
class CRow
  {
private:
   double            m_array[];
public:
   //--- конструкторы и деструктор
                     CRow(void)          { ArrayResize(m_array,0);    }
                     CRow(const CRow &r) { this=r;                    }
                     CRow(const double &array[]);
                    ~CRow(void){};
   //--- количество элементов в строке
   int               Size(void) const    { return(ArraySize(m_array));}
   //--- возвращает строку со значениями
   string            String(void) const;
   //--- оператор индексации
   double            operator[](int i) const  { return(m_array[i]);   }
   //--- операторы присваивания
   void              operator=(const double  &array[]); // массив
   void              operator=(const CRow & r);         // другой объект CRow
   double            operator*(const CRow &o);          // объект CRow для умножения
  };
//+------------------------------------------------------------------+
//| Конструктор для инициализации строки массивом                    |
//+------------------------------------------------------------------+
void  CRow::CRow(const double &array[])
  {
   int size=ArraySize(array);
//--- если массив не пустой
   if(size>0)
     {
      ArrayResize(m_array,size);
      //--- заполним значениями
```

© 2000-2025, MetaQuotes Ltd.

```
      for(int i=0;i<size;i++)
         m_array[i]=array[i];
     }
//---
  }
//+------------------------------------------------------------------+
//| Операция присваивания для массива                                |
//+------------------------------------------------------------------+
void CRow::operator=(const double &array[])
  {
   int size=ArraySize(array);
   if(size==0)
      return;
//--- заполняем массив значениями
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++)
      m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| Операция присваивания для CRow                                   |
//+------------------------------------------------------------------+
void CRow::operator=(const CRow  &r)
  {
   int size=r.Size();
   if(size==0)
      return;
//--- заполняем массив значениями
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++)
      m_array[i]=r[i];
//---
  }
//+------------------------------------------------------------------+
//| Оператор умножения на другую строку                              |
//+------------------------------------------------------------------+
double CRow::operator*(const CRow &o)
  {
   double res=0;
//--- проверки
   int size=Size();
   if(size!=o.Size() || size==0)
     {
      Print(__FUNCSIG__,": Ошибка умножения двух матриц, не совпадают размеры");
      return(res);
     }
//--- умножим поэлементно массивы и сложим произведения
   for(int i=0;i<size;i++)
      res+=m_array[i]*o[i];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- результат
   return(res);
  }
//+------------------------------------------------------------------+
//| Возвращает форматированное строковое представление               |
//+------------------------------------------------------------------+
string CRow::String(void) const
  {
   string out="";
//--- если размер массива больше нуля
   int size=ArraySize(m_array);
//--- работаем только при ненулевом кол-ве элементов в массиве
   if(size>0)
     {
      out="{";
      for(int i=0;i<size;i++)
        {
         //--- собираем значения в строку
         out+=StringFormat(" %G;",m_array[i]);
        }
      out+=" }";
     }
//--- результат
   return(out);
  }
//+------------------------------------------------------------------+
//| Класс "Матрица"                                                  |
//+------------------------------------------------------------------+
class CMatrix
  {
private:
   CRow              m_rows[];

public:
   //--- конструкторы и деструктор
                     CMatrix(void);
                     CMatrix(int rows)  { ArrayResize(m_rows,rows);             }
                    ~CMatrix(void){};
   //--- получение размеров матрицы
   int               Rows()       const { return(ArraySize(m_rows));          }
   int               Cols()       const { return(Rows()>0? m_rows[0].Size():0); }
   //--- возвращает значения столбца в виде строки CRow
   CRow              GetColumnAsRow(const int col_index) const;
   //--- возвращает строку со значениями матрицы
   string            String(void) const;
   //--- оператор индексации возвращает строку по ее номеру
   CRow *operator[](int i) const        { return(GetPointer(m_rows[i]));      }
   //--- оператор сложения
   CMatrix           operator+(const CMatrix &m);
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- оператор умножения
   CMatrix           operator*(const CMatrix &m);
   //--- оператор присваивания
   CMatrix          *operator=(const CMatrix &m);
  };
//+------------------------------------------------------------------+
//| Конструктор по умолчанию, создает массив строк нулевого размера  |
//+------------------------------------------------------------------+
CMatrix::CMatrix(void)
  {
//--- нулевое количество строк в матрице
   ArrayResize(m_rows,0);
//---
  }
//+------------------------------------------------------------------+
//| Возвращает значения столбца в виде строки CRow                   |
//+------------------------------------------------------------------+
CRow  CMatrix::GetColumnAsRow(const int col_index) const
  {
//--- переменная для получения значений из столбца
   CRow row();
//--- количество строк в матрице
   int rows=Rows();
//--- если количество строк больше нуля, выполняем операцию
   if(rows>0)
     {
      //--- массив для получения значений столбца с индексом col_index
      double array[];
      ArrayResize(array,rows);
      //--- заполнение массива
      for(int i=0;i<rows;i++)
        {
         //--- проверка номера столбца для i-ой строки на выход за пределы массива
         if(col_index>=this[i].Size())
           {
            Print(__FUNCSIG__,": Ошибка! Номер столбца ",col_index,"> размера строки "
            break; // row останется неинициализированным объектом
           }
         array[i]=this[i][col_index];
        }
      //--- создадим строку CRow на основе значений массива
      row=array;
     }
//--- результат
   return(row);
  }
//+------------------------------------------------------------------+
//| Сложение двух матриц                                             |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
CMatrix CMatrix::operator+(const CMatrix &m)
  {
//--- количество строк и столбцов в переданной матрице
   int cols=m.Cols();
   int rows=m.Rows();
//--- матрица для получения результата сложения
   CMatrix res(rows);
//--- размеры матрицы должны совпадать
   if(cols!=Cols() || rows!=Rows())
     {
      //--- нельзя произвести сложение
      Print(__FUNCSIG__,": Ошибка сложения двух матриц, не совпадают размеры");
      return(res);
     }
//--- вспомогательный массив
   double arr[];
   ArrayResize(arr,cols);
//--- перебираем строки для сложения
   for(int i=0;i<rows;i++)
     {
      //--- запишем результаты сложений строк матриц в массив
      for(int k=0;k<cols;k++)
        {
         arr[k]=this[i][k]+m[i][k];
        }
      //--- поместим массив в строку матрицы
      res[i]=arr;
     }
//--- вернем результат сложения матриц
   return(res);
  }
//+------------------------------------------------------------------+
//| Умножение двух матриц                                            |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator*(const CMatrix &m)
  {
//--- кол-во столбцов первой матрицы кол-во строк в переданной матрице
   int cols1=Cols();
   int rows2=m.Rows();
   int rows1=Rows();
   int cols2=m.Cols();
//--- матрица для получения результата сложения
   CMatrix res(rows1);
//--- матрицы должны быть согласованы
   if(cols1!=rows2)
     {
      //--- нельзя произвести умножение
      Print(__FUNCSIG__,": Ошибка умножения двух матриц, формат не согласован "
            "- число столбцов в первом сомножителе должно быть равно числу строк во в
```

© 2000-2025, MetaQuotes Ltd.

```
      return(res);
     }
//--- вспомогательный массив
   double arr[];
   ArrayResize(arr,cols1);
//--- заполняем строки в матрице произведения
   for(int i=0;i<rows1;i++)// перебираем строки
     {
      //--- обнулим массив-приемник
      ArrayInitialize(arr,0);
      //--- перебираем элементы в строке
      for(int k=0;k<cols1;k++)
        {
         //--- возьмем из матрицы m значения k-го столбца в виде строки Crow
         CRow column=m.GetColumnAsRow(k);
         //--- перемножим две строки и запишем результат скалярного умножения векторов
         arr[k]=this[i]*column;
        }
      //--- поместим массив arr[] в i-ую строку матрицы
      res[i]=arr;
     }
//--- вернем произведение двух матриц
   return(res);
  }
//+------------------------------------------------------------------+
//| Операция присваивания                                            |
//+------------------------------------------------------------------+
CMatrix *CMatrix::operator=(const CMatrix &m)
  {
//--- узнаем и зададим количество строк
   int rows=m.Rows();
   ArrayResize(m_rows,rows);
//--- заполним наши строки значениями строк переданной матрицы
   for(int i=0;i<rows;i++)
      this[i]=m[i];
//---
   return(GetPointer(this));
  }
//+------------------------------------------------------------------+
//| Строковое представление матрицы                                  |
//+------------------------------------------------------------------+
string CMatrix::String(void) const
  {
   string out="";
   int rows=Rows();
//--- формируем построчно
   for(int i=0;i<rows;i++)
      out=out+this[i].String()+"\r\n";
//--- результат
```

© 2000-2025, MetaQuotes Ltd.

```
   return(out);
  }
```

Смотри также

Перегрузка, Арифметические операции, Перегрузка функций, Приоритеты и порядок операций

© 2000-2025, MetaQuotes Ltd.

Описание внешних функций

Внешние функции, определенные в другом модуле, должны быть явно описаны. Описание включает в себя тип возвращаемого значения, имя функции и набор входных параметров с их типами. Отсутствие такого описания может привести к ошибкам при компиляции, компоновке или выполнении программы. При описании внешнего объекта используйте ключевое слово *#import* с указанием модуля.

Примеры:

```
#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
```

С помощью импорта можно очень легко описывать функции, вызываемые из внешних DLL или скомпилированных EX5 библиотек. Библиотеками EX5 являются скомпилированные ex5-файлы, которые имеют свойство library. Импортировать из EX5 библиотек можно только функции, описанные с модификатором export.

При совместном использовании DLL и EX5-библиотек следует помнить о том, что библиотеки должны иметь разные имена (вне зависимости от каталогов их размещения). Все импортируемые функции получают область видимости "имя файла" библиотеки.

Пример:

```
#import "kernel32.dll"
   int GetLastError();
#import "lib.ex5"
   int GetLastError();
#import

class CFoo
  {
public:
   int            GetLastError() { return(12345); }
   void           func()
     {
      Print(GetLastError());           // вызов метода класса
      Print(::GetLastError());         // вызов функции MQL5
      Print(kernel32::GetLastError()); // вызов функции kernel32.dll
      Print(lib::GetLastError());      // вызов функции lib.ex5
     }
  };

void OnStart()
  {
   CFoo foo;
   foo.func();
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

Смотри также

Перегрузка, Виртуальные функции, Полиморфизм

© 2000-2025, MetaQuotes Ltd.

Экспортирование функций

Существует возможность использования в mql5-программе функции, объявленной в другой mql5-программе с постмодификатором *export*. Такая функция называется экспортируемой, и она доступна для вызова из других программ после компиляции.

```
int Function() export
  {
  }
```

Данный модификатор указывает компилятору внести функцию в таблицу EX5-функций, экспортируемых данным исполняемым ex5-файлом. Только функции с таким модификатором становятся доступными ("видимыми") из других mql5-программ.

Свойство library указывает компилятору, что данный EX5-файл будет являться библиотекой, и компилятор проставит это в заголовке EX5.

Все функции, которые планируются как экспортируемые, должны быть помечены модификатором *export*.

Смотри также

Перегрузка, Виртуальные функции, Полиморфизм

© 2000-2025, MetaQuotes Ltd.

Функции обработки событий

В языке MQL5 предусмотрена обработка некоторых предопределенных событий. Функции для обработки этих событий должны быть определены в программе MQL5: имя функции, тип возвращаемого значения, состав параметров (если они есть) и их типы должны строго соответствовать описанию функции-обработчика события.

Именно по типу возвращаемого значения и по типам параметров обработчик событий клиентского терминала идентифицирует функции, обрабатывающие то или иное событие. Если у соответствующей функции указаны иные, не соответствующие нижеследующим описаниям, параметры или указан иной тип возвращаемого значения, то такая функция не будет использоваться для обработки события.

OnStart

Функция OnStart() является обработчиком события Start, которое автоматически генерируется **только** для запущенных на выполнение **скриптов**. Должна иметь тип void, параметров не имеет:

```
void OnStart();
```

Для функции OnStart() допустимо указывать тип возвращаемого значения int.

OnInit

Функция OnInit() является обработчиком события Init. Может иметь тип void или int, параметров не имеет:

```
void OnInit();
```

Событие Init генерируется сразу после загрузки эксперта или индикатора, для скриптов это событие не генерируется. Функция OnInit() используется для инициализации. Если OnInit() имеет возвращаемое значение типа int, то ненулевой код возврата означает неудачную инициализацию и генерирует событие Deinit с кодом причины деинициализации REASON_INITFAILED.

При возврате значения INIT_FAILED советник будет принудительно выгружен с графика.

При возврате значения INIT_FAILED индикатор не будет выгружен с графика. При этом оставшийся на графике индикатор является нерабочим — обработчики событий в индикаторе не вызываются.

Для оптимизации входных параметров эксперта рекомендуется в качестве кода возврата использовать значения из перечисления ENUM_INIT_RETCODE. Эти значения предназначены для организации управления ходом оптимизации, в том числе для выбора наиболее подходящих агентов тестирования. Прямо при инициализации эксперта еще до запуска самого тестирования можно запросить информацию о конфигурации и ресурсах агента (количество ядер, объем свободной памяти и т.д.) с помощью функции TerminalInfoInteger(). И на основе полученной информации либо разрешить использовать данный агент тестирования, либо отказаться от него при оптимизации данного эксперта.

ENUM_INIT_RETCODE

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
INIT_SUCCEEDED	Инициализация прошла успешно, тестирование эксперта можно продолжать. Этот код означает то же самое, что и нулевое значение - инициализация эксперта в тестере прошла успешно.
INIT_FAILED	Неудачная инициализация, тестирование нет смысла продолжать из-за неустранимых ошибок. Например, не удалось создать индикатор, необходимый для работы эксперта. Возврат этого значения означает то же самое, что и возврат значения, отличного от нуля - инициализация эксперта в тестере прошла неудачно.
INIT_PARAMETERS_INCORRECT	Предназначен для обозначения программистом некорректного набора входных параметров, в общей таблице оптимизации строка результата с таким кодом возврата будет подсвечена красным цветом. Тестирование для данного набора параметров эксперта не будет выполняться, агент свободен для получения нового задания. При получении этого значения тестер стратегий гарантированно не будет передавать данное задание другим агентам для повторного выполнения.
INIT_AGENT_NOT_SUITABLE	Ошибок в работе программы при инициализации не возникло, но по каким-то причинам данный агент не подходит для проведения тестирования. Например, недостаточно оперативной памяти, нет поддержки OpenCL и так далее. После возврата этого кода агент больше не будет получать заданий до самого конца данной оптимизации.

Функция OnInit() типа void всегда означает удачную инициализацию.

OnDeinit

Функция OnDeinit() вызывается при деинициализации и является обработчиком события Deinit. Должна быть объявлена с типом void и иметь один параметр типа const int , который содержит код причины деинициализации. Если объявлен иной тип, компилятор выдаст предупреждение, но функция вызываться не будет. Для скриптов событие Deinit не генерируется и поэтому использовать в скриптах функцию OnDeinit() нельзя.

```
void OnDeinit(const int reason);
```

Событие Deinit генерируется для экспертов и индикаторов в следующих случаях:

- перед переинициализацией в связи со сменой символа или периода графика, к которому прикреплена mql5-программа;
- перед переинициализацией в связи со сменой входных параметров;
- перед выгрузкой mql5-программы.

© 2000-2025, MetaQuotes Ltd.

OnTick

Событие NewTick генерируется **только для экспертов** при поступлении нового тика по символу, к графику которого прикреплен эксперт. Функцию OnTick() бесполезно определять в пользовательском индикаторе или скрипте, поскольку событие NewTick для них не генерируется.

Событие Tick генерируется только для экспертов, но это не означает, что эксперты обязаны иметь функцию OnTick(), так как для экспертов генерируются не только события NewTick, но и события Timer, BookEvent и ChartEvent. Должна быть объявлена с типом void, параметров не имеет:

```
void OnTick();
```

OnTimer

Функция OnTimer() вызывается при наступлении события Timer, которое генерируется системным таймером только для экспертов и индикаторов – использовать ее в скриптах нельзя. Периодичность наступления этого события устанавливается при подписке на получение функцией EventSetTimer() уведомлений о событии Timer.

Отписывание от приема посылки событий таймера для конкретного эксперта производится функцией EventKillTimer(). Функция должна быть определена с типом void, параметров не имеет:

```
void OnTimer();
```

Рекомендуется вызывать функцию EventSetTimer() однократно в функции OnInit(), а функцию EventKillTimer() вызывать однократно в OnDeinit().

Каждый эксперт и каждый индикатор работает со своим таймером, и получает события только от него. При завершении работы mql5-программы таймер уничтожается принудительно, если он был создан, но не был отключен функцией EventKillTimer().

OnTrade

Функция вызывается при наступлении события Trade, которое возникает при изменении списка выставленных ордеров и открытых позиций, истории ордеров и истории сделок. При любом торговом действии (выставлении отложенного ордера, открытии/закрытии позиции, установке стопов, срабатывании отложенных ордеров и т.п.) соответствующим образом изменяется история ордеров и сделок и/или список позиций и текущих ордеров.

```
void OnTrade();
```

Пользователь должен самостоятельно в коде реализовать проверку состояния торгового счета при получении этого события (если это необходимо по условиям торговой стратегии). Если вызов функции OrderSend() завершился успешно и вернул значение true – это означает, что торговый сервер поставил ордер в очередь на исполнение и присвоил ему номер тикета. Как только сервер обработает данный приказ, будет сгенерировано событие Trade. И если пользователь запомнил значение тикета, то при обработке события OnTrade() он может по этому тикету выяснить что именно случилось с ордером.

OnTradeTransaction

В результате выполнения определенных действий с торговым счетом, его состояние изменяется. К таким действиям относятся:

© 2000-2025, MetaQuotes Ltd.

- Отсылка торгового запроса любым MQL5-приложением в клиентском терминале при помощи функций OrderSend и OrderSendAsync и его последующее исполнение;
- Отсылка торгового запроса через графический интерфейс терминала и его последующее исполнение;
- Срабатывание отложенных ордеров и стоп-ордеров на сервере;
- Выполнение операций на стороне торгового сервера.

В результате данных действий, для счета выполняются торговые транзакции:

- обработка торгового запроса;
- изменение открытых ордеров;
- изменение истории ордеров;
- изменение истории сделок;
- изменение позиций.

Например, при отсылке рыночного ордера на покупку, он обрабатывается, для счета создается соответствующий ордер на покупку, происходит исполнение ордера, его удаление из списка открытых, добавление в историю ордеров, далее добавляется соответствующая сделка в историю и создается новая позиция. Все эти действия являются торговыми транзакциями. Приход каждой такой транзакции в терминал является событием TradeTransaction. Оно вызывает обработчик OnTradeTransaction():

```
void  OnTradeTransaction(
   const MqlTradeTransaction&    trans,        // структура торговой транзакции
   const MqlTradeRequest&        request,      // структура запроса
   const MqlTradeResult&         result        // структура ответа
   );
```

Обработчик содержит три параметра:

- **trans** - в данный параметр передается структура MqlTradeTransaction, описывающая торговую транзакцию, примененную к торговому счету;
- **request** - в данный параметр передается структура MqlTradeRequest, описывающая торговый запрос;
- **result** - в данный параметр передается структура MqlTradeResult, описывающая результат исполнения торгового запроса.

Два последних параметра **request** и **result** заполняются значениями только для транзакции типа TRADE_TRANSACTION_REQUEST, информацию о транзакции можно получить из параметра *type* переменной **trans**. Обратите внимание, что в этом случае поле *request_id* в переменной **result** содержит идентификатор торгового запроса **request**, в результате выполнения которого и была произведена торговая транзакция, описанная в переменной **trans**. Наличие идентификатора запроса позволяет связать выполненное действие (вызов функций OrderSend или OrderSendAsync) с результатом этого действия, передаваемым в OnTradeTransaction().

Один торговый запрос, отправленный из терминала вручную или через торговые функции OrderSend()/OrderSendAsync(), может порождать на торговом сервере несколько последовательных торговых транзакций. При этом очередность поступления этих транзакций в терминал не гарантирована, поэтому нельзя свой торговый алгоритм строить на ожидании поступления одних торговых транзакций после прихода других.

© 2000-2025, MetaQuotes Ltd.

- Все типы торговых транзакций описываются в перечислении ENUM_TRADE_TRANSACTION_TYPE.
- Структура MqlTradeTransaction, описывающая торговую транзакцию, заполняется по-разному в зависимости от типа транзакции. Например для транзакций типа TRADE_TRANSACTION_REQUEST необходимо анализировать только одно поле - type (тип торговой транзакции). Для получения дополнительной информации необходимо анализировать второй и третий параметры функции OnTradeTransaction (request и result). Более подробная информация приведена в разделе "Структура торговой транзакции".
- В описании торговой транзакции передается не вся доступная информация по ордерам, сделкам и позициям (например, комментарий). Для получения расширенной информации следует использовать функции OrderGet*, HistoryOrderGet*, HistoryDealGet* и PositionGet*.

После применения торговых транзакций к клиентскому счету, они последовательно помещаются в очередь торговых транзакций терминала, откуда уже последовательно передаются в точку входа OnTradeTransaction в порядке поступления в терминал.

Во время обработки торговых транзакций экспертом при помощи обработчика OnTradeTransaction(), терминал продолжает обрабатывать вновь поступающие торговые транзакции. Таким образом, состояние торгового счета может измениться уже в процессе работы OnTradeTransaction(). Например, пока MQL5-программа обрабатывает событие добавления нового ордера, он может быть исполнен, удален из списка открытых и перемещен в историю. В дальнейшем программа будет уведомлена о всех этих событиях.

Длина очереди транзакций составляет 1024 элемента. В случае, если OnTradeTransaction() будет обрабатывать очередную транзакцию слишком долго, старые транзакции в очереди могут быть вытеснены более новыми.

- В общем случае нет точного соотношения по количеству вызовов OnTrade и OnTradeTransaction. Один вызов OnTrade соответствует одному или нескольким вызовам OnTradeTransaction.
- OnTrade вызывается после соответствующих вызовов OnTradeTransaction.

OnTester

Функция OnTester() является обработчиком события Tester, которое автоматически генерируется по окончании исторического тестирования эксперта на заданном интервале дат. Функция должна быть определена с типом double, параметров не имеет:

```
double OnTester();
```

Функция вызывается непосредственно перед вызовом функции OnDeinit() и имеет тип возвращаемого значения double. Функция OnTester() может быть использована только в экспертах при тестировании и предназначена в первую очередь для расчета некоторого значения, используемого в качестве критерия Custom max при генетической оптимизации входных параметров.

При генетической оптимизации сортировка результатов в пределах одного поколения производится по убыванию. То есть, лучшими с точки зрения критерия оптимизации считаются результаты с наибольшим значением (для критерия оптимизации Custom max в расчет принимаются значения, возвращенные функцией OnTester). Худшие значения при такой

© 2000-2025, MetaQuotes Ltd.

сортировке помещаются в конец и впоследствии отбрасываются и не принимают участия в формировании следующего поколения.

OnTesterInit

Вызывается в экспертах при наступлении события TesterInit для выполнения необходимых действий перед началом оптимизации в тестере стратегий. Существуют два варианта функции.

Версия с возвратом результата

```
int  OnTesterInit(void);
```

Возвращаемое значение

Значение типа int, ноль означает успешную инициализацию эксперта, запущенного на графике перед началом оптимизации.

Приоритетным является использование вызова OnTesterInit() с возвратом результата выполнения, так как этот способ позволяет не только выполнить инициализацию программы, но и вернуть код ошибки в случае досрочного прекращения оптимизации. Возврат любого значения, отличного от INIT_SUCCEEDED (0), означает ошибку и оптимизация запущена не будет.

Версия без возврата результата оставлена только для совместимости со старыми кодами. Не рекомендуется к использованию

```
void  OnTesterInit(void);
```

Эксперт, имеющий обработчик OnTesterDeinit() или OnTesterPass(), при запуске оптимизации автоматически загружается на отдельном графике терминала с указанными в тестере символом и периодом, и получает событие TesterInit. Функция предназначена для инициализации эксперта перед началом оптимизации для последующей обработки результатов оптимизации.

OnTesterPass

Функция OnTesterPass() является обработчиком события TesterPass, которое автоматически генерируется при поступлении фрейма во время оптимизации эксперта в тестере стратегий. Функция должна быть определена с типом void, параметров не имеет:

```
void OnTesterPass();
```

Эксперт с обработчиком OnTesterPass() автоматически загружается на отдельном графике терминала с указанными для тестирования символом/периодом и получает во время оптимизации события TesterPass при получении фрейма. Функция предназначена для динамической обработки результатов оптимизации прямо "на лету", не дожидаясь её окончания. Добавление фреймов производится функцией FrameAdd(), которую можно вызывать по окончании одиночного прохода в обработчике OnTester().

OnTesterDeinit

Функция OnTesterDeinit() является обработчиком события TesterDeinit, которое автоматически генерируется по окончании оптимизации эксперта в тестере стратегий. Функция должна быть определена с типом void, параметров не имеет:

```
void OnTesterDeinit();
```

© 2000-2025, MetaQuotes Ltd.

Эксперт с обработчиком TesterDeinit() автоматически загружается на график при запуске оптимизации и получает событие TesterDeinit после её завершения. Функция предназначена для финальной обработки всех результатов оптимизации.

OnBookEvent

Функция OnBookEvent() является обработчиком события BookEvent. Событие BookEvent генерируется для экспертов и индикаторов при изменении состояния стакана цен (Depth of Market). Должна иметь тип void и один параметр типа string:

```
void OnBookEvent (const string& symbol);
```

Чтобы получать события BookEvent по любому символу, достаточно предварительно подписаться на получение этих событий для этого символа с помощью функции MarketBookAdd(). Для того чтобы отписаться от получения события BookEvent по конкретному символу, необходимо вызывать функцию MarketBookRelease().

В отличие от других событий, событие BookEvent является широковещательным. Это означает, что достаточно одному эксперту подписаться на получение события BookEvent с помощью функции MarketBookAdd, все остальные эксперты, имеющие обработчик OnBookEvent(), будут получать это событие. Поэтому необходимо анализировать имя символа, которое передается в обработчик в качестве параметра *const string& symbol*.

OnChartEvent

OnChartEvent() является обработчиком группы событий ChartEvent:

- CHARTEVENT_KEYDOWN — событие нажатия клавиатуры, когда окно графика находится в фокусе;
- CHARTEVENT_MOUSE_MOVE — события перемещения мыши и нажатия кнопок мыши (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE=true);
- CHARTEVENT_OBJECT_CREATE — событие создания графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_CREATE=true);
- CHARTEVENT_OBJECT_CHANGE — событие изменения свойств объекта через диалог свойств;
- CHARTEVENT_OBJECT_DELETE — событие удаления графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_DELETE=true);
- CHARTEVENT_CLICK — событие щелчка мыши на графике;
- CHARTEVENT_OBJECT_CLICK — событие щелчка мыши на графическом объекте, принадлежащем графику;
- CHARTEVENT_OBJECT_DRAG — событие перемещения графического объекта при помощи мыши;
- CHARTEVENT_OBJECT_ENDEDIT — событие окончания редактирования текста в поле ввода графического объекта LabelEdit;
- CHARTEVENT_CHART_CHANGE — событие изменения графика;
- CHARTEVENT_CUSTOM+n — идентификатор пользовательского события, где n находится в диапазоне от 0 до 65535.
- CHARTEVENT_CUSTOM_LAST — последний допустимый идентификатор пользовательского события (CHARTEVENT_CUSTOM+65535).

Функция может вызываться в экспертах и индикаторах, должна иметь тип void и 4 параметра:

© 2000-2025, MetaQuotes Ltd.

```
void OnChartEvent(const int id,          // идентификатор события
                  const long& lparam,    // параметр события типа long
                  const double& dparam, // параметр события типа double
                  const string& sparam   // параметр события типа string
  );
```

Для каждого типа события входные параметры функции OnChartEvent() имеют определенные значения, которые необходимы для обработки этого события. В таблице перечислены события и значения, которые передаются через параметры.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
Событие нажатия клавиатуры	CHARTEVENT_KEYDOWN	Код нажатой клавиши	Количество нажатий клавиши,сгенерированных за время её удержания в нажатом состоянии	Строковое значение битовой маски, описывающее статус кнопок клавиатуры
События мыши (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	X координата	Y координата	Строковое значение битовой маски, описывающее статус кнопок мыши
Событие создания графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Имя созданного графического объекта
Событие изменения свойств объекта через диалог свойств	CHARTEVENT_OBJECT_CHANGE	—	—	Имя измененного графического объекта
Событие удаления графического	CHARTEVENT_OBJECT_DELETE	—	—	Имя удаленного графического объекта

© 2000-2025, MetaQuotes Ltd.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
объекта (если для графика установлено свойство CHART_EVENT_OBJECT_DELETE=true)				
Событие щелчка мыши на графике	CHARTEVENT_CLICK	X координата	Y координата	—
Событие щелчка мыши на графическом объекте	CHARTEVENT_OBJECT_CLICK	X координата	Y координата	Имя графического объекта, на котором произошло событие
Событие перемещения графического объекта при помощи мыши	CHARTEVENT_OBJECT_DRAG	—	—	Имя перемещенного графического объекта
Событие окончания редактирования текста в поле ввода графического объекта "Поле ввода"	CHARTEVENT_OBJECT_ENDEDIT	—	—	Имя графического объекта "Поле ввода", в котором завершилось редактирование текста
Событие изменения графика	CHARTEVENT_CHART_CHANGE	—	—	—
Пользовательское событие с номером N	CHARTEVENT_CUSTOM+N	Значение, заданное функцией EventChartCustom()	Значение, заданное функцией EventChartCustom()	Значение, заданное функцией EventChartCustom()

OnCalculate

Функция OnCalculate() вызывается только в пользовательских индикаторах при необходимости произвести расчет значений индикатора по событию Calculate. Обычно это происходит при

© 2000-2025, MetaQuotes Ltd.

поступлении нового тика по символу, для которого рассчитывается индикатор. При этом индикатор не обязательно должен быть прикреплен к какому-нибудь ценовому графику данного символа.

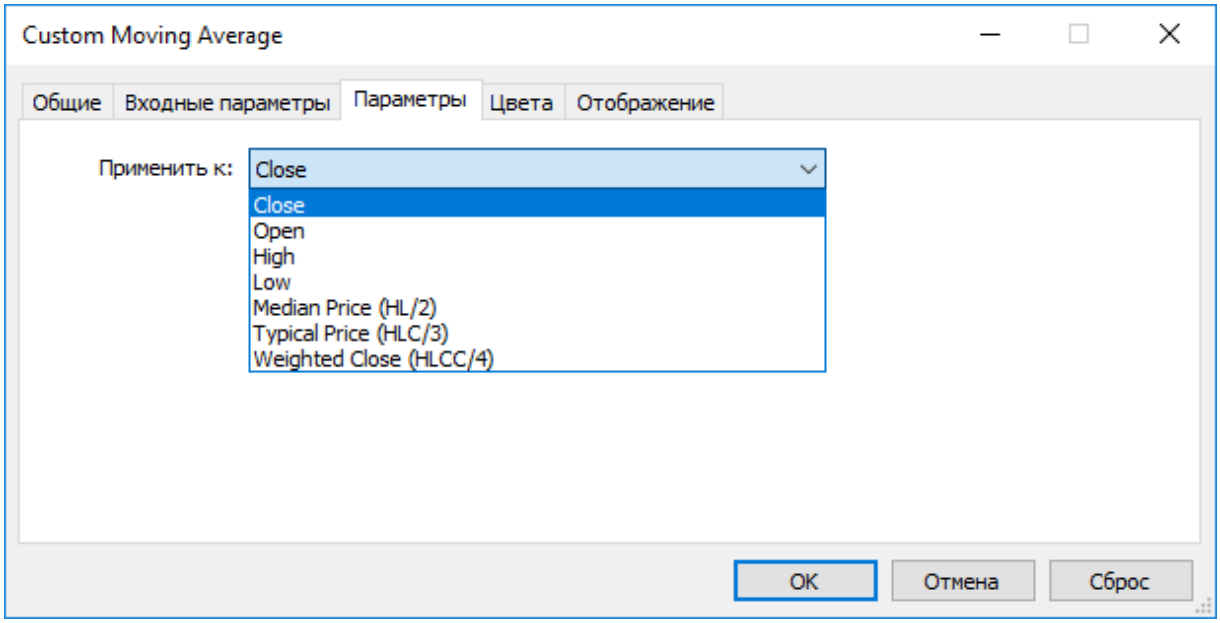
Функция OnCalculate() должна иметь тип возвращаемого значения int. Существует два варианта определения. В пределах одного индикатора нельзя использовать оба варианта функции.

Первая форма вызова предназначена для тех индикаторов, которые могут быть рассчитаны на одном буфере данных. Пример такого индикатора - Custom Moving Average.

```
int OnCalculate (const int rates_total,      // размер массива price[]
                 const int prev_calculated,  // обработано баров на предыдущем вызове
                 const int begin,            // откуда начинаются значимые данные
                 const double& price[]       // массив для расчета
   );
```

В качестве массива price[] может быть передана одна из ценовых таймсерий либо рассчитанный буфер какого-либо индикатора. Чтобы определить направление индексации в массиве price[], необходимо вызывать функцию ArrayGetAsSeries(). Чтобы не зависеть от умолчаний, необходимо безусловно вызывать функцию ArraySetAsSeries() для тех массивов, с которыми предполагается работать.

Выбор необходимой таймсерии или индикатора в качестве массива price[] осуществляется пользователем на вкладке "Parameters" при запуске индикатора. Для этого необходимо указать нужный элемент в выпадающем списке поля "Apply to".



Для получения значений пользовательского индикатора из других mql5-программ используется функция iCustom(), возвращающая хэндл индикатора для последующих операций. При этом также можно указать необходимый массив price[] или хэндл другого индикатора. Этот параметр должен передаваться последним в списке входных переменных пользовательского индикатора.

Пример:

```
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   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("Cannot open or not EX5 file '"+terminal_path+"\\MQL5\\Indicators\\"+"Cust
  }
```

В данном примере последним параметром передано значение PRICE_TYPICAL (из перечисления ENUM_APPLIED_PRICE), которое указывает, что пользовательский индикатор будет построен по типическим ценам, полученным как (High+Low+Close)/3. Если параметр не указывается, то индикатор строится по значениям PRICE_CLOSE, то есть по ценам закрытия каждого бара.

Другой пример, демонстрирующий передачу хендла индикатора последним параметром для указания массива price[], приведен в описании функции iCustom().

Вторая форма вызова предназначена для всех остальных индикаторов, у которых для расчета используется более чем одна таймсерия.

```
int OnCalculate (const int rates_total,      // размер входных таймсерий
                 const int prev_calculated,  // обработано баров на предыдущем вызове
                 const datetime& time[],     // Time
                 const double& open[],       // Open
                 const double& high[],       // High
                 const double& low[],        // Low
                 const double& close[],      // Close
                 const long& tick_volume[],  // Tick Volume
                 const long& volume[],       // Real Volume
                 const int& spread[]         // Spread
   );
```

Параметры open[], high[], low[] и close[] содержит массивы с ценами открытия, максимальной, минимальной ценами и ценами закрытия текущего таймфрейма. Параметр time[] содержит массив со значениями времени открытия, параметр spread[] - массив, содержащий историю спредов (если спред предусмотрен для данного торгового инструмента). Параметры volume[] и tick_volume[] содержат соответственно историю торгового и тикового объема.

Чтобы определить направление индексации в массивах time[], open[], high[], low[], close[], tick_volume[], volume[] и spread[], необходимо вызывать функцию ArrayGetAsSeries(). Чтобы не зависеть от умолчаний, необходимо безусловно вызывать функцию ArraySetAsSeries() для тех массивов, с которыми предполагается работать.

Первый параметр rates_total содержит количество баров, доступных индикатору для расчета, и соответствует количеству баров, доступных на графике.

Необходимо отметить связь между значением, возвращаемым функцией OnCalculate() и вторым входным параметром prev_calculated. Параметр prev_calculated при вызове функции содержит значение, которое **вернула** функция OnCalculate() на **предыдущем** вызове. Это позволяет реализовать экономные алгоритмы расчета пользовательского индикатора с тем, чтобы избежать повторных расчетов для тех баров, которые не изменились с предыдущего запуска этой функции.

© 2000-2025, MetaQuotes Ltd.

Для этого обычно достаточно вернуть значение параметра rates_total, которое содержит количество баров при текущем вызове функции. Если с момента последнего вызова функции OnCalculate() ценовые данные были изменены (подкачана более глубокая история или были заполнены пропуски истории), то значение входного параметра prev_calculated будет установлено в нулевое значение самим терминалом.

Примечание: если функция OnCalculate возвращает нулевое значение, то в окне DataWindow клиентского терминала значения индикатора не показываются.

Для лучшего понимания будет полезно запустить индикатор, код которого приложен ниже.

Пример индикатора:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Line
#property indicator_label1  "Line"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDarkBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         LineBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- получим количество доступных баров для текущих символа и периода на графике
   int bars=Bars(Symbol(),0);
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("Bars = ",bars,", rates_total = ",rates_total,", prev_calculated = ",prev_cal
   Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Смотри также

Выполнение программ, События клиентского терминала, Работа с событиями

© 2000-2025, MetaQuotes Ltd.

Переменные

Объявление переменных

Переменные должны быть объявлены перед их использованием. Для идентификации переменных используются уникальные имена. Описания переменных используются для их определения и объявления типов. Описание не является оператором.

Простыми типами являются:

- char, short, int, long, uchar, ushort, uint, ulong - целые числа;
- color - целое число, представляющее RGB-цвет;
- datetime - дата и время, беззнаковое целое число, содержащее количество секунд, прошедших с 0 часов 1 января 1970 года;
- bool - логические значения true и false;
- double - числа двойной точности с плавающей точкой;
- float - числа одинарной точности с плавающей точкой;
- string - символьные строки.

Примеры:

```
string szInfoBox;
int    nOrders;
double dSymbolPrice;
bool   bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color    cModify_Color = C'0x44,0xB9,0xE6';
```

Сложные или составные типы:

Структуры - это составные типы данных, построенные с использованием других типов.

```
struct MyTime
  {
   int hour;    // 0-23
   int minute;  // 0-59
   int second;  // 0-59
  };
...
MyTime strTime; // переменная типа объявленной ранее структуры MyTime
```

До тех пор пока структура не объявлена, нельзя объявлять переменные типа структуры.

Массивы

Массив - это индексированная совокупность однотипных данных:

```
int    a[50];       // Одномерный массив из 50 целых чисел.
double m[7][50];    // Двухмерный массив из семи массивов,
                    // каждый из которых состоит из 50 чисел.
```

© 2000-2025, MetaQuotes Ltd.

```
MyTime t[100];       // массив содержащий элементы типа MyTime
```

Индексом массива может быть только целое число. Допускаются не более чем четырехмерные массивы. Нумерация элементов массива начинается с 0. Последний элемент одномерного массива имеет номер на 1 меньший, чем размер массива, то есть обращение к последнему элементу массива из 50 целых чисел будет выглядеть как a[49]. То же самое относится и к многомерным массивам - индексация одного измерения производится от 0 до размер измерения-1. Последний элемент двумерного массива из примера будет выглядеть как m[6][49].

Статические массивы не могут быть представлены в виде таймсерий, то есть к ним не применима функция ArraySetAsSeries(), которая устанавливает доступ к элементам массива от конца массива к его началу. Если требуется обеспечить доступ к массиву как в таймсериях, используйте объект динамического массива.

При доступе за пределы массива исполняющая подсистема сгенерирует критическую ошибку и выполнение программы будет остановлено.

Встроенные методы для работы с массивами

Для работы с массивами можно использовать функции из раздела Операции с массивами, а также встроенные методы:

Метод	Аналог	Описание
void array.Fill(const scalar value, const int start_pos=0, const int count=-1);	ArrayFill, ArrayInitialize	Заполняет массив указанным значением
void array.Free();	ArrayFree	Освобождает буфер динамического массива и устанавливает размер нулевого измерения в 0 (ноль)
int array.Resize(const int range0_size, const int reserve); int array.Resize(const int range_sizes[], const int reserve);	ArrayResize	Устанавливает новый размер в первом измерении массива
int array.Print();	ArrayPrint	Выводит в журнал значения массива простого типа
int array.Size(const int range=-1);	ArraySize, ArrayRange	Возвращает количество элементов во всем массиве (range=-1) либо в указанном измерении массива
bool array.IsDynamic();	ArrayIsDynamic	Проверяет, является ли массив динамическим
bool array.IsIndicatorBuffer();		Проверяет, является ли массив индикаторным буфером
bool array.IsSeries();	ArrayIsSeries	Проверяет, является ли массив таймсерией

© 2000-2025, MetaQuotes Ltd.

Метод	Аналог	Описание
bool array.AsSeries();	ArrayGetAsSeries	Проверяет направление индексации массива
bool array.AsSeries(const bool as_series);	ArraySetAsSeries	Устанавливает направление индексации в массиве
int array.Copy(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCopy	Копирует значения массива в другой массив
int array.Compare(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCompare	Возвращает результат сравнения двух массивов простых типов или пользовательских структур
int array.Insert(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayInsert	Вставляет в массив-приемник из массива-источника указанное число элементов
int array.Remove(const int start_pos, const int count);	ArrayRemove	Удаляет из массива указанное число элементов начиная с указанного индекса
int array.Reverse(const int start_pos, const int count);	ArrayReverse	Разворачивает в массиве указанное число элементов начиная с указанного индекса
bool array.Swap(array& arr[]);	ArraySwap	Обменивается содержимым с другим динамическим массивом такого же типа
void array.Sort(sort_function);	ArraySort	Сортировка числовых массивов по первому измерению
int array.Search(scalar value, search_function);	ArrayBsearch	Возвращает индекс первого найденного элемента в первом измерении массива
int array.Find((scalar value, search_function);		Производит поиск в массиве с помощью переданной функции и возвращает индекс первого найденного элемента
array array.Select(scalar value, search_function);		Производит поиск в массиве с помощью переданной функции и возвращает массив со всеми найденными элементами

Спецификаторы доступа

Спецификаторы доступа указывают компилятору каким образом можно осуществлять доступ к переменным, членам структур или классов.

© 2000-2025, MetaQuotes Ltd.

Спецификатор const объявляет переменную константой, и не позволяет изменять значение этой переменной в процессе выполнения программы. Допускается однократная инициализация переменной при ее объявлении.

Пример

```
int OnCalculate (const int rates_total,      // размер массива price[]
                 const int prev_calculated,  // обработано баров на предыдущем вызове
                 const int begin,            // откуда начинаются значимые данные
                 const double& price[]       // массив для расчета
   );
```

Для доступа к членам структур и классов используются следующие спецификаторы:

- public - разрешает ничем неограниченный доступ к переменной или методу класса;
- protected - разрешает доступ со стороны методов данного класса, а также со стороны методов публично наследуемых классов. Иной доступ невозможен;
- private - разрешает доступ к переменным и методам класса только из методов данного класса.
- virtual - применим только к методам класса (но не к методам структур) и сообщает компилятору, что данный метод должен быть размещен в таблице виртуальных функций класса.

Классы памяти

Существуют три класса памяти: static, input и extern. Эти модификаторы класса памяти явно указывают компилятору, что соответствующие переменные распределяются в предопределенной области памяти, называемой глобальным пулом. При этом данные модификаторы указывают на особую обработку данных переменных.

Если переменная, объявленная на локальном уровне, не является статической, то распределение памяти под такую переменную производится автоматически на программном стеке. Освобождение памяти, выделенной под не статический массив, производится также автоматически при выходе за пределы области видимости блока, в котором массив объявлен.

Смотри также

Типы данных, Инкапсуляция и расширяемость типов, Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов, Статические члены класса

© 2000-2025, MetaQuotes Ltd.

Локальные переменные

Переменная, объявленная внутри какой-либо функции, является локальной. Область видимости локальной переменной ограничена пределами функции, внутри которой она объявлена. Локальная переменная может быть проинициализирована при помощи любого выражения. Инициализация локальной переменной производится каждый раз при вызове соответствующей функции. Локальные переменные располагаются во временной области памяти соответствующей функции.

Пример:

```
int somefunc()
  {
   int ret_code=0;
   ...
   return(ret_code);
  }
```

Область действия (или область видимости) переменной - это часть программы, в которой на переменную можно сослаться. Переменные, объявленные внутри блока (на внутреннем уровне), имеют областью действия блок. Область действия блок начинается объявлением переменной и заканчивается конечной правой фигурной скобкой.

Локальные переменные, объявленные в начале функции, имеют область действия блок так же, как и параметры функции, являющиеся локальными переменными. Любой блок может содержать объявления переменных. Если блоки вложены и идентификатор во внешнем блоке имеет такое же имя, как идентификатор во внутреннем блоке, идентификатор внешнего блока "невидим" (скрыт) до момента завершения работы внутреннего блока.

Пример:

```
void OnStart()
  {
//---
   int i=5;       // локальная переменная функции
     {
      int i=10;  // переменная функции
      Print("В блоке i = ",i); // результат  i = 10;
     }
   Print("Вне блока i = ",i);  // результат  i = 5;
  }
```

Это означает, что пока выполняется внутренний блок, он видит значения своих собственных локальных идентификаторов, а не значения идентификаторов с идентичными именами в охватывающем блоке.

Пример:

```
void OnStart()
  {
//---
   int i=5;       // локальная переменная функции
   for(int i=0;i<3;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("Внутри for i = ",i);
   Print("Вне блока i = ",i);
  }
/* Результат выполнения
Внутри for i = 0
Внутри for i = 1
Внутри for i = 2
Вне блока i = 5
*/
```

Локальные переменные, объявленные как static, имеют областью действия блок, несмотря на то, что они существуют с самого начала выполнения программы.

Стек

В каждой MQL5-программе под хранение локальных автоматически создаваемых переменных функций выделяется специальная область памяти, называемая стеком. Стек выделяется один на все функции и по умолчанию для индикаторов его размер равен 1Mb. В экспертах и скриптах размером стека можно управлять директивой компилятора #property stacksize (задает размер стека в байтах), для них по умолчанию под стек выделяется 8Mb.

Статические локальные переменные размещаются там же, где и другие статические и глобальные переменные, в специальной области памяти, существующей отдельно от стека. Динамически создаваемые переменные также используют отдельную от стека область памяти.

При каждом вызове функции для внутренних нестатических переменных отводится место на стеке. При выходе из функции память становится доступной для повторного использования.

Если из первой функции производится вызов второй функции, то та в свою очередь занимает на стеке необходимый объем под свои переменные из оставшейся стековой памяти. Таким образом при вложенных вызовах функций на стеке будет заниматься память последовательно под каждую функцию. Это может привести к нехватке памяти при очередном вызове функции, такая ситуация называется переполнением стека.

Поэтому для больших локальных данных лучше использовать динамическую память - при входе в функцию память под локальные нужды выделять в системе (new, ArrayResize()), а при выходе из функции производить освобождение памяти (delete, ArrayFree()).

Смотри также

Типы данных, Инкапсуляция и расширяемость типов, Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Формальные параметры

Передаваемые в функцию параметры являются локальными. Областью видимости является блок функции. Формальные параметры должны отличаться по именам от внешних переменных и локальных переменных, определенных внутри функции. В блоке функции формальным параметрам могут быть присвоены некоторые значения. Если же формальный параметр объявлен с модификатором const, то его значение не может быть изменено внутри функции.

Пример:

```
void func(const int & x[], double y, bool z)
  {
   if(y>0.0 && !z)
      Print(x[0]);
   ...
  }
```

Формальные параметры могут быть проинициализированы константами. В этом случае инициализирующее значение считается значением по умолчанию. Параметры, следующие за проинициализированным параметром, должны быть тоже проинициализированы.

Пример:

```
void func(int x, double y = 0.0, bool z = true)
  {
   ...
  }
```

При вызове такой функции инициализированные параметры можно опускать, вместо них будут подставлены значения по умолчанию.

Пример:

```
func(123, 0.5);
```

Параметры простых типов передаются по значению, то есть изменения соответствующей локальной переменной такого типа внутри вызываемой функции никак не отразятся в вызывающей функции. Массивы любых типов и данные типа структур всегда передаются по ссылке. Если необходимо запретить изменение массива или содержимого структуры, параметры этих типов должны быть объявлены с ключевым словом const.

Существует возможность передавать по ссылке и параметры простых типов. В этом случае модификация таких параметров внутри вызываемой функции отразится на соответствующих переменных, переданных по ссылке. Для того чтобы указать, что параметр передается по ссылке, после типа данных необходимо поставить модификатор &.

Пример:

```
void func(int& x, double& y, double & z[])
  {
   double calculated_tp;
   ...
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<OrdersTotal(); i++)
     {
      if(i==ArraySize(z))
         break;
      if(OrderSelect(i)==false)
         break;
      z[i]=OrderOpenPrice();
     }
   x=i;
   y=calculated_tp;
  }
```

Параметры, передаваемые по ссылке, нельзя инициализировать значениями по умолчанию.

В функцию нельзя передать больше 64 параметров.

Смотри также

Input переменные, Типы данных, Инкапсуляция и расширяемость типов, Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Статические переменные

Класс памяти static определяет статическую переменную. Модификатор static указывается перед типом данных.

Пример:

```
int somefunc()
  {
   static int flag=10;
   ...
   return(flag);
  }
```

Статическая переменная может быть проинициализирована соответствующей ее типу константой или константным выражением, в отличие от простой локальной переменной, которая может быть проинициализирована любым выражением.

Статические переменные существуют с момента выполнения программы и инициализируются однократно перед вызовом специализированной функции OnInit(). Если начальные значения не указаны, то переменные статического класса памяти принимают нулевые начальные значения.

Локальные переменные, объявленные с ключевым словом static сохраняют свои значения в течение всего времени существования функции. При каждом следующем вызове функции такие локальные переменные содержат те значения, которые они имели при предыдущем вызове.

Любые переменные в блоке, кроме формальных параметров функции, могут быть определены как статические. Если переменная, объявленная на локальном уровне не является статической, то распределение памяти под такую переменную производится автоматически.

Пример:

```
int Counter()
  {
   static int count;
   count++;
   if(count%100==0)
      Print("Функция Counter была вызвана уже ",count," раз");
   return
      count;
  }
void OnStart()
  {
//---
   int c=345;
   for(int i=0;i<1000;i++)
      int c=Counter();
   Print("c = ",c);
  }
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

Типы данных, Инкапсуляция и расширяемость типов, Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов, Статические члены класса

© 2000-2025, MetaQuotes Ltd.

Глобальные переменные

Глобальные переменные создаются путем размещения их объявлений вне описания какой-либо функции. Глобальные переменные определяются на том же уровне, что и функции, т. е. не локальны ни в каком блоке.

Пример:

```
int GlobalFlag=10;    // глобальная переменная
int OnStart()
  {
   ...
  }
```

Область видимости глобальных переменных - вся программа, глобальные переменные доступны из всех функций, определенных в программе. Инициализируются нулем, если явно не задано другое начальное значение. Глобальная переменная может быть проинициализирована только соответствующей ее типу константой либо константным выражением.

Инициализация глобальных переменных производится однократно после загрузки программы в память клиентского терминала и перед первой обработкой события Init. Для глобальных переменных, представляющих собой объекты классов, при инициализации вызываются соответствующие конструкторы. В скриптах инициализация глобальных переменных производится перед обработкой события Start.

Замечание: не следует путать переменные, объявленные на глобальном уровне, с глобальными переменными клиентского терминала, доступ к которым осуществляется при помощи функций GlobalVariable...().

Смотри также

Типы данных, Инкапсуляция и расширяемость типов, Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Input переменные

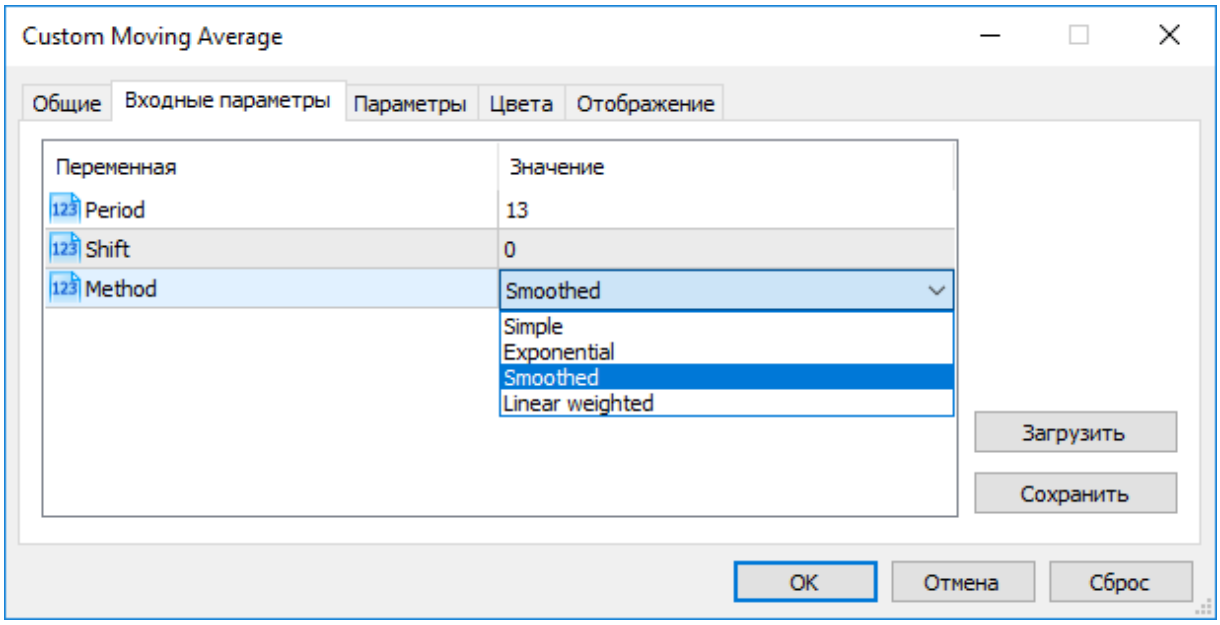
Класс памяти input определяет внешнюю переменную. Модификатор input указывается перед типом данных. Изменять значение переменной с модификатором input внутри mql5-программы нельзя, такие переменные доступны только для чтения. Изменять значения input-переменных может только пользователь из окна свойств программы. Внешние переменные всегда переинициализируются непосредственно перед вызовом OnInit().

Максимальная длина имени input-переменных составляет 63 символа. При этом для входного параметра типа string **максимальная** длина значения (длина строки) может иметь размер от 191 до 253 символов (смотри Примечание). Минимальная длина - 0 символов (значение не задано).

Пример:

```
//--- input parameters
input int             MA_Period=13;
input int             MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMA;
```

Input переменные определяют входные параметры программы, они доступны из окна свойств программы.



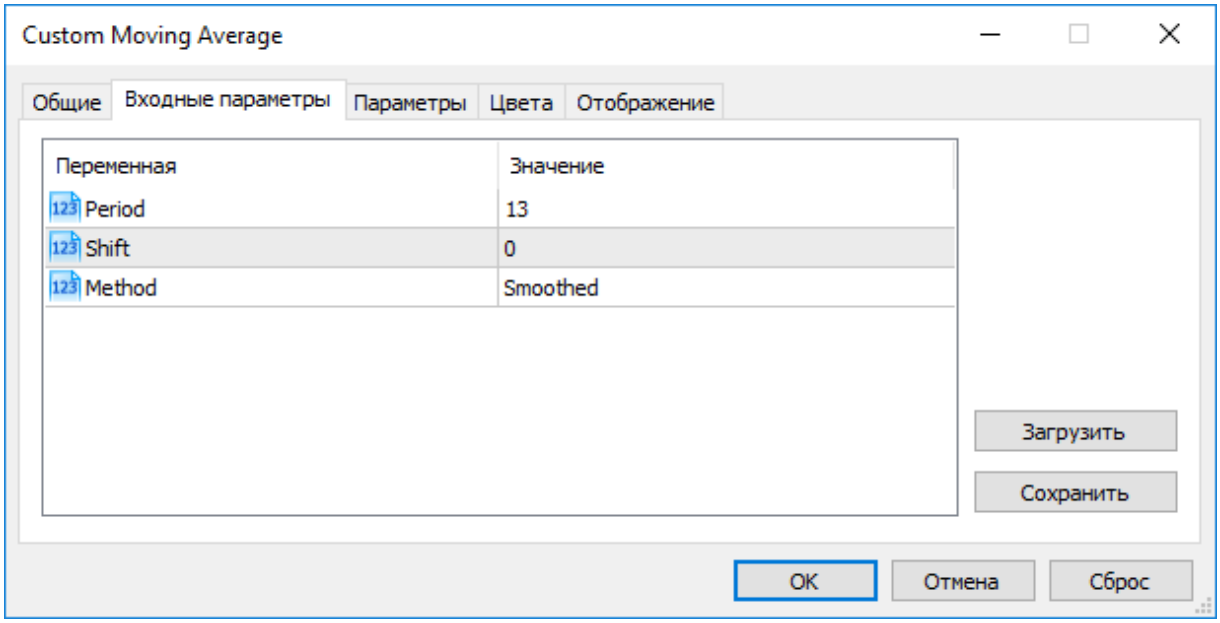
Существует возможность задать иной способ отображения имен входных параметров на закладке "Inputs". Для этого используется строчный комментарий, который должен располагаться после описания входного параметра в той же строке. Таким образом, входным параметрам можно сопоставить более понятные для пользователя имена.

Пример:

```
//--- input parameters
input int             InpMAPeriod=13;         // Smoothing period
input int             InpMAShift=0;           // Line horizontal shift
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_MA_METHOD InpMAMethod=MODE_SMMA;  // Smoothing method
```



Примечание: Массивы и переменные сложных типов не могут выступать в качестве input-переменных.

Примечание: Длина строчного комментария для Input переменных не может превышать 63 символа.

Примечание: Для input-переменных типа string ограничение на длину значения (длину строки) задается следующими условиями:

- значение параметра представлется строкой "имя_параметра=значение_параметра"(символ '=' учитывается),
- максимальная длина представления 255 символов (*total_length_max*=255 или 254 символа без учета '='),
- максимальная длина имени строкового параметра *parameter_name_length* = 63 символа.

Таким образом, максимальный размер строки для string-параметра можно вычислить по формуле:

```
parameter_value_length=total_length_max-parameter_name_length=254-parameter_name_lengt
```

Это дает максимальный размер строки от 191 (parameter_name_length=63) до 253 символов (parameter_name_length=1).

Передача параметров при вызове пользовательских индикаторов из mql5-программ

Пользовательские индикаторы вызываются при помощи функции iCustom(). При этом после имени пользовательского индикатора должны идти параметры в точном соответствии с объявлением input-переменных данного пользовательского индикатора. Если параметров указывается меньше, чем объявлено input-переменных в вызываемом пользовательском индикаторе, то недостающие параметры заполняются значениями, указанными при объявлении переменных.

© 2000-2025, MetaQuotes Ltd.

Если в пользовательском индикаторе используется функция OnCalculate первого вида (то есть, индикатор считается на одном массиве данных), то в качестве последнего параметра при вызове такого пользовательского индикатора должно выступать одно из значений ENUM_APPLIED_PRICE либо хэндл другого индикатора. При этом все параметры, соответствующие input-переменным, должны быть явно указаны.

Перечисления в качестве input-параметра

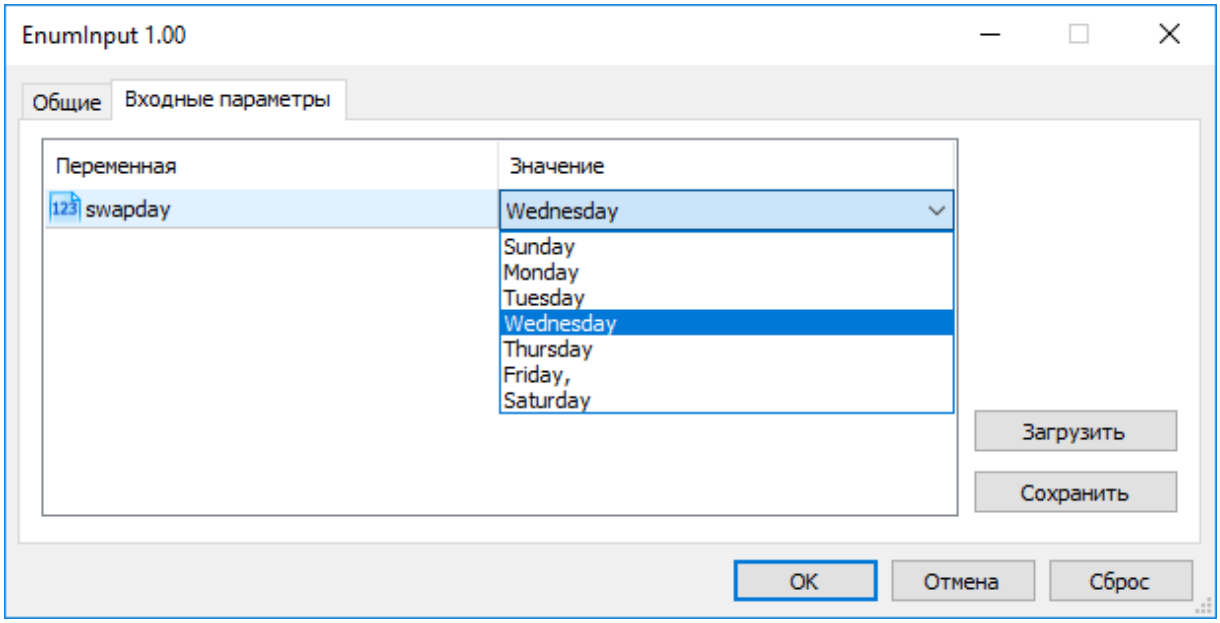
В качестве input-переменных (входных параметров для mql5-программ) можно использовать не только предусмотренные языком MQL5 встроенные перечисления, но и перечисления, заданные пользователем. Например, мы можем создать перечисление dayOfWeek, описывающее дни недели, и использовать input-переменную для указания конкретного дня недели не в виде цифры, а в более привычном для пользователя виде.

Пример:

```
#property script_show_inputs
//--- day of week
enum dayOfWeek
  {
   S=0,      // Sunday
   M=1,      // Monday
   T=2,      // Tuesday
   W=3,      // Wednesday
   Th=4,     // Thursday
   Fr=5,     // Friday,
   St=6,     // Saturday
  };
//--- input parameters
input dayOfWeek swapday=W;
```

Для того чтобы при запуске скрипта пользователь мог выбрать нужное значение из окна свойств, мы используем команду препроцессора #property script_show_inputs. Запускаем скрипт на исполнение и можем выбрать из списка одно из значений перечислений dayOfWeek. Запускаем скрипт EnumInInput и переходим на закладку "Параметры". По умолчанию, значение параметра swapday (день начисления тройного свопа) является среда (W=3), но мы можем задать любое другое значение и использовать это значение для изменения работы программы.

© 2000-2025, MetaQuotes Ltd.



Количество возможных значений перечисления ограничено. Поэтому для выбора входного значения используется выпадающий список. В качестве значений, показываемых в списке, используются мнемонические имена членов перечисления. Если же мнемоническому имени сопоставлен комментарий, как это показано в нашем примере, то вместо мнемонического имени используется содержимое комментария.

Каждое значение из перечисления dayOfWeek имеет свое значение от 0 до 6, но в списке параметров будут показаны комментарии, указанные для каждого значения. Это дает дополнительную гибкость для написания программ с понятными описаниями входных параметров.

Переменные с модификатором sinput

Переменные с модификатором input позволяют не только задавать значения внешних параметров при запуске программ, но также играют большую роль при оптимизации торговых стратегий в тестере. Каждая объявленная в эксперте input-переменная, за исключением типа string, может участвовать в оптимизации.

В некоторых случаях бывает необходимо исключить некоторые внешние параметры программы из формирования области всех возможных проходов в тестере. Специально для таких случаев существует модификатор памяти sinput. sinput - это сокращенное написание объявления статической внешней переменной: sinput = static input. То есть такое объявление в коде советника

```
sinput       int layers=6;    // Количество слоев
```

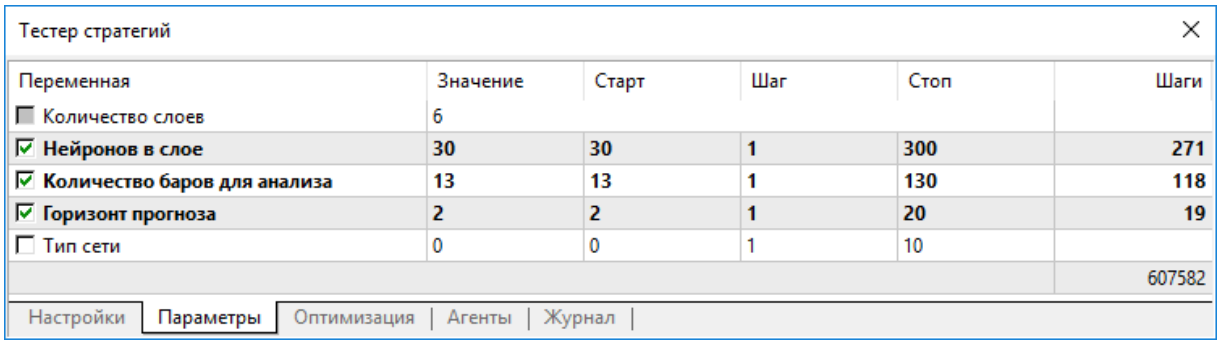
будет эквивалентно полному объявлению

```
static input int layers=6;    // Количество слоев
```

Переменная, объявленная с модификатором sinput, является входным параметром MQL5-программы, значение этого параметра можно изменять при её запуске. Но при этом данная

© 2000-2025, MetaQuotes Ltd.

переменная не участвует в процессе оптимизации входных параметров, то есть не производится перебор её значений при поиске наилучшего набора параметров по заданному критерию.



Переменная	Значение	Старт	Шаг	Стоп	Шаги
Количество слоев	6				
Нейронов в слое	**30**	**30**	**1**	**300**	**271**
Количество баров для анализа	**13**	**13**	**1**	**130**	**118**
Горизонт прогноза	**2**	**2**	**1**	**20**	**19**
Тип сети	0	0	1	10	
					607582

На рисунке показано, что эксперт имеет 5 внешних параметров, из них параметр "Количество слоев" объявлен как sinput и равен 6. Этот параметр не может изменяться в процедуре оптимизации торговой стратегии, для него возможно установить нужное значение, которое и будет использоваться. Поля Старт, Шаг и Стоп для такой переменной не доступны для установки значений.

Таким образом, задав для переменной модификатор sinput, мы запрещаем пользователю оптимизировать данный параметр. Это значит, что в тестере стратегий пользователю терминала становится недоступным задавать для неё начальное и конечное значения для автоматического перебора в указанном диапазоне в процессе оптимизации.

Но при этом есть одно исключение из данного правила – sinput-переменные можно варьировать в задачах оптимизации с помощью функции ParameterSetRange(). Данная функция создана специально для программного управления пространством доступных значений для любых input-переменных, в том числе и объявленных как static input (sinput). Другая функция ParameterGetRange() позволяет при запуске оптимизации (в обработчике OnTesterInit()) получить значения input-переменных и в случае необходимости переопределить шаг изменения и диапазон, в пределах которого будет перебираться значение оптимизируемого параметра.

Таким образом, сочетание модификатора sinput и двух функций по работе с input-параметрами позволяет создавать гибкие правила для задания интервалов оптимизации одних input-переменных в зависимости от значения других input-переменных.

Группировка входных параметров

Для удобства работы с MQL5-программами входные параметры можно разбивать на именованные блоки с помощью ключевого слова group. Это позволяет визуально отделить одни параметры от других на основе заложенной в них логики.

```
input group             "Имя группы"
input int               variable1 = ...
input double            variable2 = ...
input double            variable3= ...
```

Все входные параметры после такого объявления визуально объединяются в указанную группу, что позволяет упростить для пользователя MQL5-программы настройку параметров при запуске на

© 2000-2025, MetaQuotes Ltd.

графике или в тестере стратегий. Указание каждой группы действует до появления нового объявления группы:

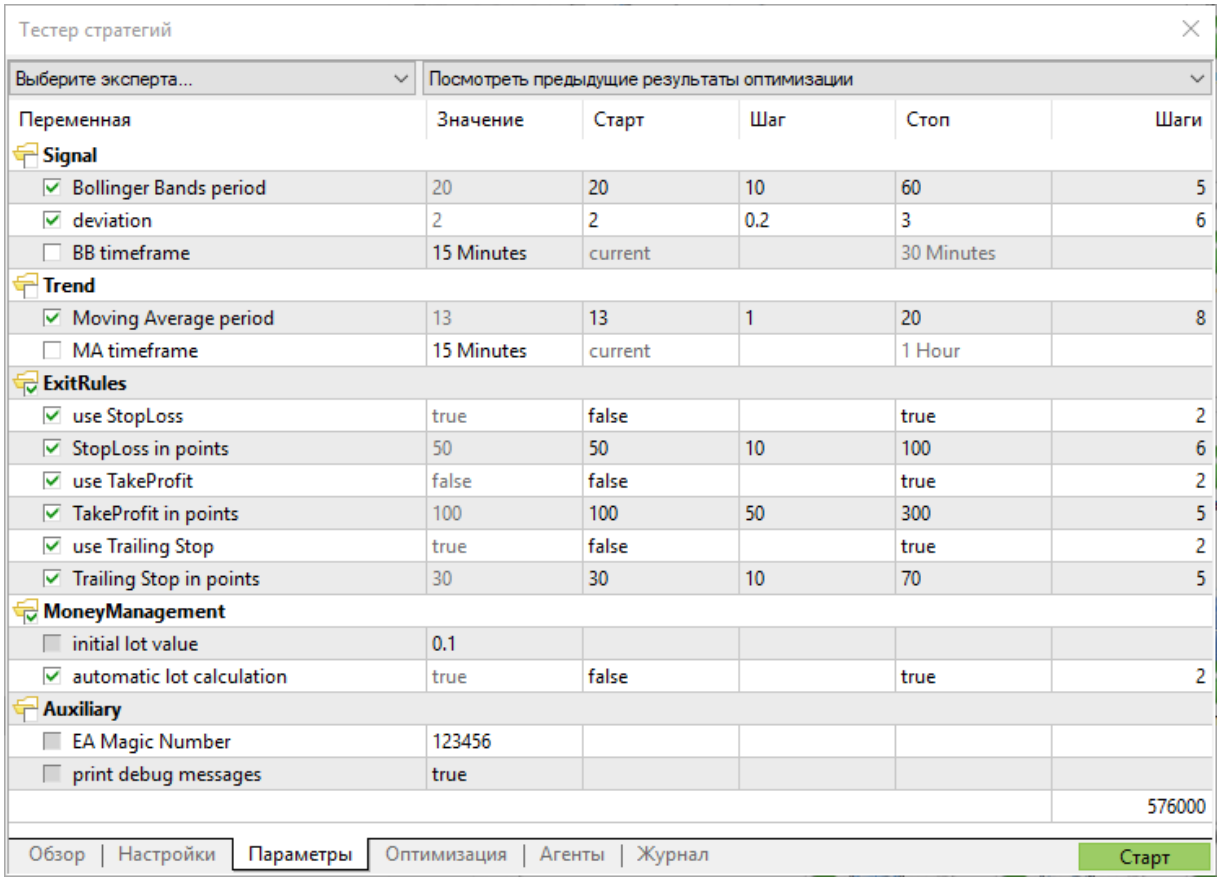
```
input group           "Имя группы #1"
input int             group1_var1 = ...
input double          group1_var2 = ...
input double          group1_var3 = ...

input group           "Имя группы #2
input int             group2_var1 = ...
input double          group2_var2 = ...
input double          group2_var3 = ...
```

Пример советника, у которого блок входных параметров разбит по их назначению:

```
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
```

При запуске такого советника в тестере стратегий появляется возможность двойным кликом по имени группы сворачивать и разворачить блок входных параметров, а также одним кликом по чекбоксу группы выделять все её параметры для оптимизации.

© 2000-2025, MetaQuotes Ltd.



Смотри также

iCustom, Перечисления, Свойства программ

© 2000-2025, MetaQuotes Ltd.

Extern переменные

Ключевое слово extern используется, чтобы объявить идентификаторы переменных как идентификаторы статического класса памяти с глобальным временем жизни. Такие переменные существуют с момента начала выполнения программы и для них память выделяется и инициализируется сразу после начала выполнения программы.

Можно создавать программы, которые состоят из нескольких исходных файлов, для этого используется директива препроцессору #include. Переменные, объявленные как extern с одним и тем же типом и идентификатором, могут существовать в разных исходных файлах одного проекта.

При компиляции всего проекта все extern-переменные с одним и тем же типом и идентификатором ассоциируются с одним участком памяти пула глобальных переменных. Extern-переменные полезны для раздельной компиляции исходных файлов. Extern-переменные можно инициализировать, но только однократно – недопустимо существование нескольких инициализированных extern-переменных одного и того же типа и с одним и тем же идентификатором.

Смотри также

Типы данных, Инкапсуляция и расширяемость типов, Инициализация переменных, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Инициализация переменных

Любая переменная при определении может быть инициализирована. Если не произведена явная инициализация переменной, то значение, хранящееся в данной переменной, может быть каким угодно. Неявная инициализация не производится.

Глобальные и статические переменные могут быть проинициализированы только константой соответствующего типа или константным выражением. Локальные переменные могут быть проинициализированы любым выражением, а не только константой.

Инициализация глобальных и статических переменных производится однократно. Инициализация локальных переменных производится каждый раз при вызове соответствующих функций.

Примеры:

```
int    n       = 1;
string s       = "hello";
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 } };
//--- из тетриса
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                 WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- инициализация всех полей структуры нулевым значением
MqlTradeRequest request={};
```

Список значений элементов массива должен быть заключен в фигурные скобки. Пропущенные инициализирующие последовательности считаются равными 0.

Если размер инициализируемого массива не указан, то он определяется компилятором, исходя из размера инициализирующей последовательности.

Примеры:

```
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},{0},100};
   str10 s10_3={{1,0},{1.0}};
//---
   Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
```

© 2000-2025, 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]);
  }
```

Для переменных типа структур допускается частичная инициализация, это же относится и к статическим массивам (с явно заданным размером). Можно проинициализировать один или несколько первых элементов структуры или массива, оставшиеся элементы в таком случае будут проинициализированы нулями.

Смотри также

Типы данных, Инкапсуляция и расширяемость типов, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Область видимости и время жизни переменных

Существует два основных вида области видимости: локальная область видимости и глобальная область видимости.

Переменная, объявленная вне всех функций, помещается в глобальную область видимости. Доступ к таким переменным может осуществляться из любого места программы. Такие переменные располагаются в глобальном пуле памяти, поэтому время их жизни совпадает со временем жизни программы.

Переменная, объявленная внутри блока (часть кода, заключенная в фигурные скобки), принадлежит локальной области видимости. Такая переменная не видна (поэтому и недоступна) за пределами блока, в котором она объявлена. Самый распространенный случай локального объявления - переменная, объявленная внутри функции. Переменная, объявленная локально, располагается на стеке, и время жизни такой переменной совпадает со временем жизни функции.

Так как областью видимости локальной переменной является блок, в котором она объявлена, то существует возможность объявлять переменные с именем, совпадающим с именами переменных, объявленных в других блоках; а также объявленных на более верхних уровнях, вплоть до глобального.

Пример:

```
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;
      //--- set empty value for first limit bars
      for(i=0; i<limit; i++)
         LineBuffer[i]=0.0;
      //--- calculate first visible value
      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++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      sum=0;
      for(int j=0; j<MA_Period; j++)
         sum+=(MA_Period-j)*price[i-j];
      LineBuffer[i]=sum/weightsum;
     }
//---
  }
```

Обратите внимание на переменную i, объявленную в строке

```
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
```

Ее область видимости – только цикл for, за пределами этого цикла действует другая переменная с тем же именем, объявленная в начале функции. Кроме того, в теле цикла объявлена переменная k, областью видимости которой является тело цикла.

Локальные переменные можно объявлять со спецификатором доступа static. В этом случае компилятор располагает такую переменную в глобальном пуле памяти. Поэтому, время жизни статической переменной совпадает со временем жизни программы. При этом область видимости такой переменной ограничивается пределами блока, в котором она объявлена.

Смотри также

Типы данных, Инкапсуляция и расширяемость типов, Инициализация переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Создание и уничтожение объектов

После загрузки на исполнение mql5-программы каждой переменной выделяется память в соответствие с типом переменной. Переменные делятся на два типа по уровню доступа - глобальные переменные и локальные переменные, и по классам памяти: входные параметры mql5-программы, статические и автоматические. Каждая переменная при необходимости инициализируется соответствующим значением. После использования переменная деинициализируется и память, использованная ею, возвращается исполняемой системе MQL5.

Инициализация и деинициализация глобальных переменных

Инициализация глобальных переменных производится автоматически сразу после загрузки mql5-программы и до вызова любой функции. При инициализации производится присвоение начальных значений переменным простых типов и вызывается конструктор для объектов, если он есть. Входные переменные всегда объявляются на глобальном уровне, инициализируются значениями, задаваемыми пользователями в диалоге при запуске mql5-программы.

Несмотря на то, что статические переменные обычно объявляются на локальном уровне, память под эти переменные распределяется заранее, и инициализация производится сразу после загрузки программы, точно так же как и для глобальных переменных.

Порядок инициализации соответствует порядку объявления переменной в программе, а деинициализация производится в обратном порядке перед выгрузкой mql5-программы. Это правило только для тех переменных, которые не были созданы оператором new. Такие переменные создаются и инициализируются автоматически сразу после загрузки, а деинициализируются непосредственно перед выгрузкой программы.

Инициализация и деинициализация локальных переменных

Если переменная, объявленная на локальном уровне, не является статической, то распределение памяти под такую переменную производится автоматически. Локальные переменные, также как и глобальные, инициализируются автоматически в тот момент, когда выполнение программы встречает объявление локальной переменной. Таким образом, порядок инициализации соответствует порядку объявления.

Локальные переменные деинициализируются в конце блока программы, в котором они объявлены, и в порядке, обратном их объявлению. Блок программы – это составной оператор, который может являться частью оператора выбора switch, цикла(for, while, do-while), телом функции или частью оператора if-else.

Инициализация локальных переменных происходит только в тот момент, когда выполнение программы доходит до объявления переменной. Если в процессе выполнения программы блок, в котором объявлена переменная, не был выполнен, то такая переменная не инициализируется.

Инициализация и деинициализация динамически размещаемых объектов

Особый случай представляют из себя указатели объектов, так как объявление указателя не влечет за собой инициализации соответствующего объекта. Динамически размещаемые объекты инициализируются только в момент создания экземпляра класса оператором new. Инициализация

© 2000-2025, MetaQuotes Ltd.

объекта предполагает вызов конструктора соответствующего класса. Если соответствующего конструктора в классе нет, то его члены, имеющие простой тип, не будут автоматически проинициализированы; члены типов строка, динамический массив и сложный объект будут автоматически проинициализированы.

Указатели могут быть объявлены на локальном или глобальном уровне и при этом могут быть проинициализированы пустым значением NULL или значением указателя такого же или порожденного типа. Если для указателя, объявленного на локальном уровне, был вызван оператор *new*, то и оператор *delete* для этого указателя должен быть выполнен до выхода из этого уровня. В противном случае указатель будет потерян, и объект не сможет быть удален явно.

Все объекты, созданные выражением *указатель_объекта*=new *Имя_Класса*, обязательно должны быть впоследствии уничтожены оператором delete(*указатель_объекта*). Если по каким то причинам такая переменная по окончании работы программы не была уничтожена оператором delete, то об этом будет выведено сообщение в журнал "Эксперты". Можно объявить несколько переменных и всем им присвоить указатель одного объекта.

Если динамически создаваемый объект имеет конструктор, то этот конструктор будет вызван в момент выполнения оператора *new*. Если объект имеет деструктор, то деструктор будет вызван в момент выполнения оператора *delete*.

Таким образом, динамически размещаемые объекты создаются только в момент создания оператором *new* и гарантированно уничтожаются либо оператором *delete*, либо автоматически исполняющей системой MQL5 в момент выгрузки программы. Порядок объявления указателей динамически создаваемых объектов не влияет на порядок их инициализации. Порядок инициализации и деинициализации полностью контролируется программистом.

Особенности работы с динамической памятью

При работе с динамическими массивами освобожденная память сразу же возвращается в систему.

При создании динамического объекта класса через new, память сначала ищется в пуле памяти классов, с которым работает менеджер памяти, и если в пуле недостаточно памяти, то память запрашивается в системе. При удалении динамического объекта через delete, память, занимаемая объектом, возвращается в пул памяти классов.

Менеджер памяти возвращает память в систему сразу после выхода из функций-обработчиков событий: OnInit(), OnDeinit(), OnStart(), OnTick(), OnCalculate(), OnTimer(), OnTrade(), OnTester(), OnTesterInit(), OnTesterPass(), OnTesterDeinit(), OnChartEvent(), OnBookEvent().

Краткая характеристика переменных

Основные сведения о порядке создания, уничтожении, вызове конструкторов и деструкторов приведены в краткой таблице.

	Глобальная автоматическая переменная	Локальная автоматическая переменная	Динамически создаваемый объект
Инициализация	сразу после загрузки mql5-программы	при достижении в ходе выполнения	при выполнении оператора **new**

© 2000-2025, MetaQuotes Ltd.

	Глобальная автоматическая переменная	Локальная автоматическая переменная	Динамически создаваемый объект
		строки кода, где она объявлена	
Порядок инициализации	в порядке объявления	в порядке объявления	не зависит от порядка объявления
Деинициализация	перед выгрузкой mql5-программы	при выходе выполнения из блока объявления	при выполнении оператора **delete** или перед выгрузкой mql5-программы
Порядок деинициализации	в порядке, обратном инициализации	в порядке, обратном инициализации	не зависит от порядка инициализации
Вызов конструктора	при загрузке mql5-программы	при инициализации	при выполнении оператора *new*
Вызов деструктора	при выгрузке mql5-программы	при выходе из блока, в котором переменная была инициализирована	при выполнении оператора *delete*
Сообщения об ошибках	сообщение в журнал "Эксперты" о попытке удаления автоматически созданного объекта	сообщение в журнал "Эксперты" о попытке удаления автоматически созданного объекта	сообщение в журнал "Эксперты" о неудаленных динамически созданных объектах при выгрузке mql5-программы

Смотри также

Типы данных, Инкапсуляция и расширяемость типов, Инициализация переменных, Область видимости и время жизни переменных

© 2000-2025, MetaQuotes Ltd.

Препроцессор

Препроцессор - это специальная подсистема компилятора MQL5, которая занимается предварительной подготовкой исходного текста программы непосредственно перед ее компиляцией.

Препроцессор позволяет улучшить читаемость исходного кода. Структурирование кода может быть достигнуто путем включения отдельных файлов с исходными кодами mql5-программ. Улучшению читаемости кода способствует и возможность присвоения мнемонических имен отдельным константам.

Препроцессор позволяет также определять специфические параметры mql5-программ:

- Объявлять константы
- Устанавливать свойства программы
- Включать в текст программы файлы
- Импортировать функции
- Использовать условную компиляцию

Директивы препроцессора используются компилятором для предварительной обработки исходного кода перед его компиляцией. Директива всегда начинается со знака # (решетка), поэтому компилятор запрещает использовать данный символ в именах переменных, функций и т.д.

Каждая директива описывается отдельной записью и действует до переноса строки. Нельзя в одной записи использовать несколько директив. Если запись директивы получается слишком большой, то её можно разбить на несколько строк с помощью обратного слеша '\', в таком случае следующая строка будет считаться продолжением записи директивы.

```
//+------------------------------------------------------------------+
//|  псевдооператор 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"};
//--- обход массива с помощью ForEach
   ForEach(i,array)
     {
      PrintFormat("%d: array[%d]=%s",i,i,array[i]);
     }
  }
//+------------------------------------------------------------------+
 /* Результат вывода
   0: array[0]=12
   1: array[1]=23
```

© 2000-2025, MetaQuotes Ltd.

```
   2: array[2]=34
   3: array[3]=45
*/
```

Для компилятора все эти три строчки директивы #define будут выглядеть как одна длинная строка. В этом примере также используется двойной знак решетки ##, который называется оператором слияния и используется в макросах #define для объединения двух токенов макроса в один. Оператор слияния токенов не может быть первым или последним в определении макроса.

© 2000-2025, MetaQuotes Ltd.

Макроподстановка (#define, #undef)

Директивы препроцессора используются компилятором для предварительной обработки исходного кода перед его компиляцией. Директива всегда начинается со знака # (решетка), поэтому компилятор запрещает использовать данный символ в именах переменных, функций и т.д.

Каждая директива описывается отдельной записью и действует до переноса строки. Нельзя в одной записи использовать несколько директив. Если запись директивы получается слишком большой, то её можно разбить на несколько строк с помощью обратного слеша '\', в таком случае следующая строка будет считаться продолжением записи директивы.

Директива #define может быть использована для присвоения мнемонических имен выражениям. Существует две формы:

```
#define identifier expression                    // беспараметрическая форма
#define identifier(par1,... par8) expression     // параметрическая форма
```

Директива #define подставляет expression вместо всех последующих найденных вхождений identifier в исходном тексте. identifier заменяется только в том случае, если он представляет собой отдельный токен. identifier не заменяется, если он является частью комментария, частью строки, или частью другого более длинного идентификатора.

Идентификатор константы подчиняется тем же правилам, что и для имен переменных. Значение может быть любого типа:

```
#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");
  }
```

expression может состоять из нескольких токенов, таких как ключевые слова, константы, константные и неконстантные выражения. expression заканчивается с концом строки и не может быть перенесено на следующую строку.

Пример:

```
#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);
  }
/* Результат
   2 + 3*2 = 8
```

© 2000-2025, MetaQuotes Ltd.

```
   (2 + 3)*2 = 10
*/
```

Параметрическая форма #define

При параметрической форме все последующие найденные вхождения identifier будут заменены на expression с учетом фактических параметров. Например,

```
 // пример с двумя параметрами a и b
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
выражение double c=MUL(A,B);
равносильно double c=((2+3)*(5-1));
*/
// Результат
// c=20
```

Обязательно заключайте параметры в круглые скобки при использовании параметров в expression, так как это позволит избежать неочевидных ошибок, которые трудно найти. Если переписать пример без использования скобок, то результат окажется совсем другой:

```
 // пример с двумя параметрами a и b
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
выражение double c=MUL(A,B);
равносильно double c=2+3*5-1;
*/
// Результат
// c=16
```

При использовании параметрической формы допускается не более 8 параметров.

```
 // правильная параметрическая форма
#define LOG(text)  Print(__FILE__,"(",__LINE__,") :",text)   // один параметр - 'text'

 // неправильная параметрическая форма
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9)   p1+p2+p3+p4 // более 8 параме
```

© 2000-2025, MetaQuotes Ltd.

Директива #undef

Директива #undef предназначена для отмены макроса, объявленного ранее.

Пример:

```
#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();
  }

/* Результат:
 MACRO is defined in func1
 MACRO is not defined in func2
*/
```

Смотри также

Идентификаторы, Символьные константы

© 2000-2025, MetaQuotes Ltd.

Свойства программ (#property)

У каждой mql5-программы можно указать дополнительные специфические параметры *#property*, которые помогают клиентскому терминалу правильно обслуживать программы без необходимости их явного запуска. В первую очередь это касается внешних настроек индикаторов. Свойства, описанные во включаемых файлах, полностью игнорируются. Свойства необходимо задавать в главном mq5-файле.

```
#property идентификатор значение
```

Компилятор запишет в настройках выполняемого модуля объявленные значения.

Константа	Тип	Описание
icon	string	Путь к файлу с картинкой, которая будет показываться для программы EX5. Правила указания пути такие же, как и для ресурсов. Свойство должно указываться в главном модуле с исходным кодом MQL5. Файл иконки должен быть в формате ICO.
link	string	Ссылка на сайт компании-производителя
copyright	string	Название компании-производителя
version	string	Версия программы, не более 31 символа
description	string	Краткое текстовое описание mql5-программы. Может присутствовать несколько description, каждый из которых описывает одну строку текста. Общая длина всех description не может превышать 511 символов с учетом переводов строк
stacksize	int	Указывает размер стека для MQL5 программы, стек достаточного объема требуется в случае выполнения рекурсивных вызовов функций. При запуске скрипта или эксперта на графике выделяется стек не менее 8Мб, для индикаторов свойство не работает - стек всегда фиксированного объема в 1Мб. При запуске в тестере программе всегда выделяется стек в размере 16 Мб.
library		Библиотека; не назначается никакой стартовой функции, функции с модификатором export можно импортировать в других mql5-программах
indicator_applied_price	int	Задает значение по умолчанию для поля "Apply to". Можно задавать одно из значений перечисления ENUM_APPLIED_PRICE. Если свойство не задано, то по умолчанию применяется значение PRICE_CLOSE

© 2000-2025, MetaQuotes Ltd.

Константа	Тип	Описание
indicator_chart_window		Выводить индикатор в окно графика
indicator_separate_window		Выводить индикатор в отдельное окно
indicator_height	int	Фиксированная высота подокна индикатора в пикселях (свойство INDICATOR_HEIGHT)
indicator_buffers	int	Количество буферов для расчета индикатора
indicator_plots	int	Количество графических серий в индикаторе
indicator_minimum	double	Нижнее ограничение шкалы отдельного окна индикатора
indicator_maximum	double	Верхнее ограничение шкалы отдельного окна индикатора
indicator_labelN	string	Задает метку для N-ой графической серии, отображаемую в окне DataWindow. Для графических серий, требующих несколько индикаторных буферов (DRAW_CANDLES, DRAW_FILLING и другие), имена меток задаются через разделитель ';'.
indicator_colorN	color	Цвет для вывода линии N, где N - номер графической серии; нумерация с 1
indicator_widthN	int	Толщина линии в графической серии, где N - номер графической серии; нумерация с 1
indicator_styleN	int	Стиль линии в графической серии, указываемый с помощью значения из ENUM_LINE_STYLE. N - номер графической серии, нумерация с 1
indicator_typeN	int	Вид графического построения, указываемый с помощью значения из ENUM_DRAW_TYPE. N - номер графической серии, нумерация с 1
indicator_levelN	double	Горизонтальный уровень N в отдельном окне индикатора
indicator_levelcolor	color	Цвет горизонтальных уровней индикатора
indicator_levelwidth	int	Толщина горизонтальных уровней индикатора
indicator_levelstyle	int	Стиль горизонтальных уровней индикатора
script_show_confirm		Выводить окно подтверждения перед запуском скрипта
script_show_inputs		Выводить окно со свойствами перед запуском скрипта и запретить вывод окна подтверждения
tester_indicator	string	Имя пользовательского индикатора в формате "имя_индикатора.ex5". Необходимые для тестирования индикаторы определяются

© 2000-2025, MetaQuotes Ltd.

Константа	Тип	Описание
		автоматически из вызова функций iCustom(), если соответствующий параметр задан константной строкой. Для остальных случаев (использование функции IndicatorCreate() или использование неконстантной строки в параметре, задающем имя индикатора) необходимо данное свойство
tester_file	string	Имя файла для тестера с указанием расширения, заключенное в двойные кавычки (как константная строка). Указанный файл будет передан тестеру в работу. Входные файлы для тестирования, если необходимы, должны указываться всегда
tester_library	string	Имя библиотеки с расширением, заключенное в двойные кавычки. Библиотека может быть как с расширением dll, так и с расширением ex5. Необходимые для тестирования библиотеки определяются автоматически. Однако если какая-либо библиотека используется пользовательским индикатором, то необходимо использовать данное свойство
tester_set	string	Имя set-файла со значениями и шагом входных параметров. Указанный файл будет передан тестеру в работу перед началом тестирования или оптимизации. Имя файла необходимо указывать с расширением и в двойных кавычках как константную строку. Если в названии set-файла указать имя эксперта и номер версии как "<expert_name>_<number>.set", то он автоматически добавится в меню загрузки версий параметров под номером версии <number>. Например, имя "MACD Sample_4.set" означает, что это set-файл для эксперта "MACD Sample.mq5" с номером версии равным 4. Для изучения формата рекомендуем вручную сохранить настройки тестирования/оптимизации в тестере стратегий и затем открыть созданный таким образом set-файл.
tester_no_cache	string	Тестер стратегий при выполнении оптимизации сохраняет все результаты выполненных проходов в кеш оптимизации, в котором для каждого набора входных параметров сохраняется результат тестирования. Это позволяет при

© 2000-2025, MetaQuotes Ltd.

Константа	Тип	Описание
		повторной оптимизации на тех же параметрах брать готовые результаты без повторного вычисления и затрат времени. Но для некоторых задач – например, при математических вычислениях – может потребоваться проводить расчеты независимо от наличия готовых результатов в кеше оптимизации. В этом случае в файл необходимо включить свойство *tester_no_cache*. При этом сами результаты тестирования все равно будут сохраняться в кеше, чтобы можно было в тестере стратегий посмотреть все данные по выполненным проходам.
tester_everytick_calculate	string	В тестере стратегий индикаторы рассчитываются только при обращении к ним за данными – то есть только в тот момент, когда запрашиваются значения индикаторных буферов. Это даёт существенное ускорение при тестировании и оптимизации, если не требуется получать значения индикатора на каждом тике. Указание свойства *tester_everytick_calculate* позволяет при тестировании принудительно включить режим расчета индикатора на каждом тике. Индикаторы в тестере стратегий также принудительно считаются на каждом тике в следующих случаях: • при тестировании в визуальном режиме; • при наличии в индикаторе функций EventChartCustom, OnChartEvent, OnTimer; • если индикатор создан компилятором с номером билда ниже 1916. Данное свойство касается только работы в тестере стратегий, в терминале индикаторы всегда считаются на каждом поступившем тике.
optimization_chart_mode	string	Указывает тип графика и имена двух входных параметров, которые будут использоваться для визуализации результатов оптимизации. Например, "3d, InpX, InpY" означает, что будет показан 3D-график с координатными осями на основе значений перебираемых параметров InpX и InpY. Таким образом, с помощью данного

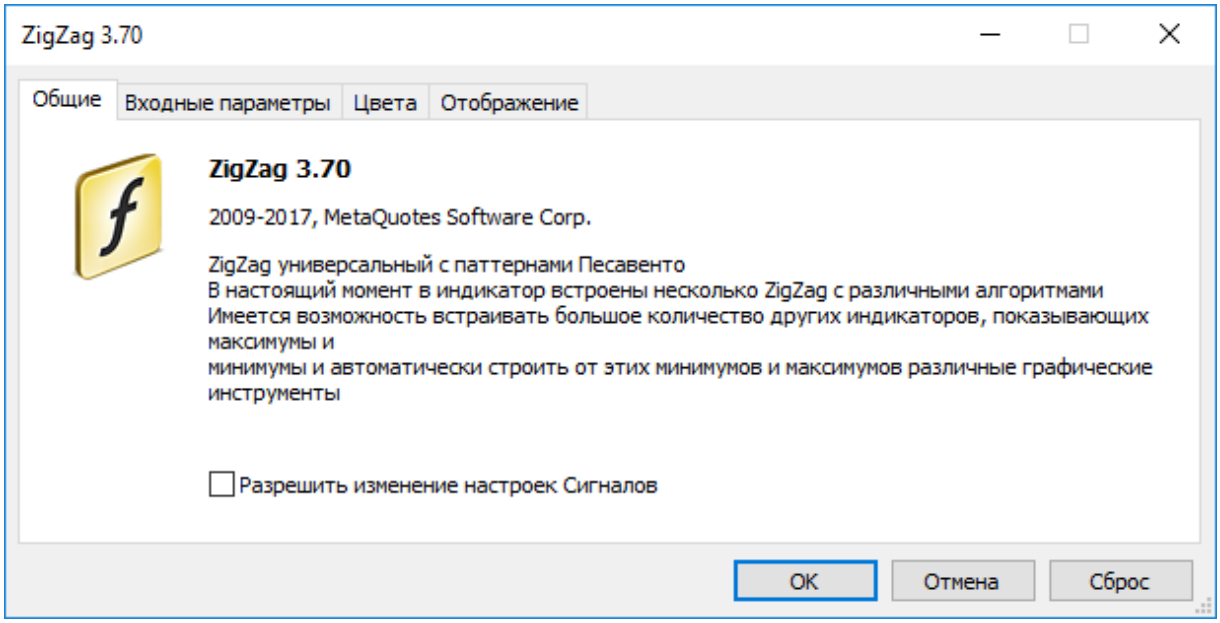
© 2000-2025, MetaQuotes Ltd.

Константа	Тип	Описание
		свойства можно прямо в коде указывать параметры, которые будут использоваться для показа графика оптимизации и тип самого графика. Возможные варианты: • "3d, input_parameter_name1, input_parameter_name2" - объемный 3D-график визуализации, который можно вращать, приближать, отдалять. Строится по двум входным параметрам. • "2d, input_parameter_name1, input_parameter_name2" - плоский график в виде сетки, каждая клетка которого закрашена цветом в зависимости от результата. Строится по двум входным параметрам. • "1d, input_parameter_name1, input_parameter_name2" - линейный график, в котором результаты отсортированы по указанному параметру. Каждый проход отбражается в виде точки, строится по одному входному параметру. • "0d, input_parameter_name1, input_parameter_name2" - обычный график с результатами, отсортированными в порядке поступления результата прохода. Каждый проход отбражается в виде точки. Не требует указания параметров, но указанные параметры будут использоваться при ручном переключении на другой тип графика. Можно указать только тип графика и не указывать один или два входных параметра - терминал в таком случае сам выберет необходимые параметры для показа графика оптимизации.

Пример задания описания и номера версии

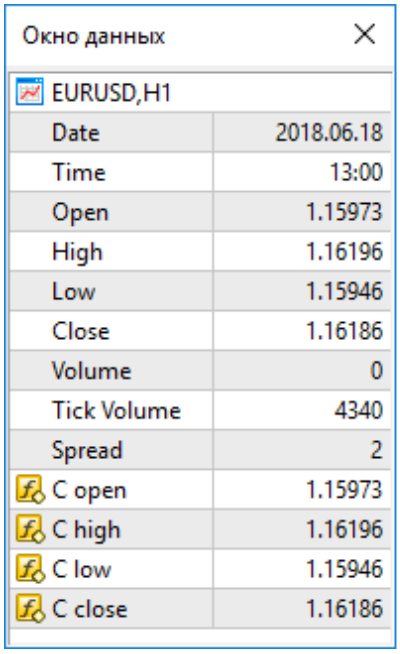
```
#property version     "3.70"      // текущая версия эксперта
#property description "ZigZag универсальный с паттернами Песавенто"
#property description "В настоящий момент в индикатор встроены несколько ZigZag с разл
#property description "Имеется возможность встраивать большое количество других индика
#property description "минимумы и автоматически строить от этих минимумов и максимумов
```

© 2000-2025, MetaQuotes Ltd.



Пример указания отдельной метки для каждого индикаторного буфера ("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"
```





© 2000-2025, MetaQuotes Ltd.

Включение файлов (#include)

Командная строка *#include* может встречаться в любом месте программы, но обычно все включения размещаются в начале файла исходного текста. Формат вызова:

```
#include <имя_файла>
#include "имя_файла"
```

Примеры:

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

Препроцессор заменяет строку *#include <имя_файла>* содержимым файла WinUser32.mqh. Угловые скобки обозначают, что файл WinUser32.mqh будет взят из стандартного каталога (обычно это *каталог_терминала\MQL5\Include*). Текущий каталог не просматривается.

Если имя файла заключено в кавычки, то поиск производится в текущем каталоге (в котором содержится основной файл исходного текста). Стандартный каталог не просматривается.

Смотри также

Стандартная библиотека, Импорт функций

© 2000-2025, MetaQuotes Ltd.

Импорт функций (#import)

Импорт функций осуществляется из откомпилированных модулей MQL5 (файлы *.ex5) и из модулей операционной системы (файлы *.dll). Имя модуля указывается в директиве *#import*. Для того чтобы компилятор мог правильно оформить вызов импортируемой функции и организовать правильную передачу параметров, необходимо полное описание функций. Описания функций следуют непосредственно за директивой *#import "имя модуля"*. Новая команда *#import* (можно без параметров) завершает блок описания импортируемых функций.

```
#import "имя_файла"
    func1 define;
    func2 define;
    ...
    funcN define;
#import
```

Импортируемые функции могут иметь любые имена. Можно одновременно импортировать из разных модулей функции с одинаковыми именами. Импортируемые функции могут иметь имена, совпадающие с именами встроенных функций. Операция разрешения контекста определяет, какая из функций должна вызываться.

Порядок поиска файла, указанного после ключевого слова #import, описан в разделе Вызов импортируемых функций.

Так как импортируемые функции находятся вне компилируемого модуля, компилятор не может проверить правильность передаваемых параметров. Поэтому, во избежание ошибок выполнения, необходимо точно описывать состав и порядок параметров, передаваемых в импортируемые функции. Параметры, передаваемые в импортируемые функции (как из EX5, так и из DLL-модулей), могут иметь значения по умолчанию.

В импортируемых функциях в качестве параметров нельзя использовать:

- указатели (*);
- ссылки на объекты, содержащие динамические массивы и/или указатели.

В импортируемые из DLL функции нельзя передавать в качестве параметра классы, массив строк или сложные объекты, содержащие строки и/или динамические массивы любых типов.

Примеры:

```
#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);
```

© 2000-2025, MetaQuotes Ltd.

```
#import
```

Для импорта функций во время выполнения mql5-программы используется раннее связывание. Это значит, что библиотека загружается в процессе загрузки использующей ее ex5-программы.

Не рекомендуется использовать полностью квалифицированное имя загружаемого модуля вида *Drive:\Directory\FileName.Ext*. Библиотеки MQL5 загружаются из папки *terminal_dir\MQL5\Libraries*.

Если импортируемая функция имеет разные варианты вызова для 32-х и 64-х битной версий Windows, то необходимо импортировать оба и явно вызывать нужный вариант функции с помощью переменной _IsX64.

Пример:

```
#import "user32.dll"
//--- для 32-х битной системы
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
//--- для 64-х битной системы
int    MessageBoxW(ulong hWnd,string lpText,string lpCaption,uint uType);
#import
//+------------------------------------------------------------------+
//|  MessageBox_32_64_bit использует нужный вариант  MessageBoxW()   |
//+------------------------------------------------------------------+
int MessageBox_32_64_bit()
  {
   int res=-1;
   //--- если у нас 64-битная Windows
   if(_IsX64)
     {
      ulong hwnd=0;
      res=MessageBoxW(hwnd,"Пример вызова 64-битной версии MessageBoxW","MessageBoxW 6
     }
   else  // у нас 32-битная Windows
     {
      uint hwnd=0;
      res=MessageBoxW(hwnd,"Пример вызова 32-битной версии MessageBoxW","MessageBoxW 3
     }
   return (res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int ans=MessageBox_32_64_bit();
   PrintFormat("MessageBox_32_64_bit returned %d",ans);
  }
```

© 2000-2025, MetaQuotes Ltd.

Импорт функций из .NET библиотек

Для работы с функциями .NET библиотеки достаточно импортировать саму DLL без указания конкретных функций. MetaEditor автоматически импортирует все функции, с которыми возможна работа:

- Простые структуры (POD, plain old data) - структуры, которые содержат только простые типы данных.
- Публичные статические функции, в параметрах которых используются только простые типы и структуры POD или их массивы

Для вызова функций из библиотеки просто импортируйте ее:

```
#import "TestLib.dll"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int x=41;
   TestClass::Inc(x);
   Print(x);
  }
```

C# код функции Inc класса TestClass выглядит следующим образом:

```
public class TestClass
  {
   public static void Inc(ref int x)
      {
       x++;
      }
  }
```

В результате выполнения скрипт вернет значение 42.

Смотри также

Включение файлов

© 2000-2025, MetaQuotes Ltd.

Условная компиляция (#ifdef, #ifndef, #else, #endif)

Директивы препроцессора используются компилятором для предварительной обработки исходного кода перед его компиляцией. Директива всегда начинается со знака # (решетка), поэтому компилятор запрещает использовать данный символ в именах переменных, функций и т.д.

Каждая директива описывается отдельной записью и действует до переноса строки. Нельзя в одной записи использовать несколько директив. Если запись директивы получается слишком большой, то её можно разбить на несколько строк с помощью обратного слеша '\', в таком случае следующая строка будет считаться продолжением записи директивы.

Директивы условной компиляции препроцессора позволяют компилировать или пропускать часть программы в зависимости от выполнения некоторого условия.

Условие может принимать одну из описываемых ниже форм.

```
#ifdef identifier
   // код, находящийся здесь, компилируется, если identifier уже был определен для пре
#endif
#ifndef identifier
   // код, находящийся здесь, компилируется, если identifier в данный момент не опреде
#endif
```

За любой из команд условной компиляции может следовать произвольное число строк, содержащих, возможно, команду вида #else и заканчивающихся #endif. Если проверяемое условие справедливо, то строки между #else и #endif игнорируются. Если же проверяемое условие не выполняется, то игнорируются все строки между проверкой и командой #else, а если ее нет, то командой #endif.

Пример:

```
#ifndef TestMode
   #define TestMode
#endif
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef TestMode
      Print("Test mode");
   #else
      Print("Normal mode");
   #endif
  }
```

В зависимости от типа программы и режима компиляции стандартные макросы определяются следующим образом:

Макрос __MQL5__ доступен при компиляции файла *.mq5, при компиляции *.mq4 доступен макрос __MQL4__.

© 2000-2025, MetaQuotes Ltd.

Макрос _DEBUG доступен при компиляции под отладку.
Макрос _RELEASE доступен при компиляции не под отладку.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef __MQL5__
      #ifdef _DEBUG
         Print("Hello from MQL5 compiler [DEBUG]");
      #else
        #ifdef _RELEASE
           Print("Hello from MQL5 compiler [RELEASE]");
        #endif
      #endif
   #else
      #ifdef __MQL4__
        #ifdef _DEBUG
           Print("Hello from MQL4 compiler [DEBUG]");
        #else
           #ifdef _RELEASE
              Print("Hello from MQL4 compiler [RELEASE]");
           #endif
        #endif
      #endif
   #endif
  }
```

© 2000-2025, MetaQuotes Ltd.

Объектно-ориентированное программирование

Объектно-ориентированное программирование – это программирование, сфокусированное на данных, причем данные и поведение неразрывно связаны между собой. Вместе данные и поведение представляют собой класс, а объекты являются экземплярами класса.

Составными частями объектно-ориентированного подхода являются:

- Инкапсуляция и расширяемость типов
- Наследование
- Полиморфизм
- Перегрузка
- Виртуальные функции

ООП рассматривает вычисления как моделирование поведения. То, что моделируется, является объектами, представленными вычислительными абстракциями. Допустим, мы хотим написать хорошо всем известную игру "Тетрис", для этого мы должны научиться моделировать появление случайной фигуры, составленной из четырех квадратиков, соединенных друг с другом ребрами. Также требуется регулировать скорость падения фигуры, задать операции вращения и сдвига фигуры. Перемещения фигуры на экране ограничены границами стакана, это требование мы также должны смоделировать. Кроме того, заполненные ряды кубиков в стакане должны уничтожаться и необходимо вести подсчет очков, заработанных в игре.

Таким образом, такая простая в понимании игра требует создания нескольких моделей - модель фигуры, модель стакана, модель движения фигуры в стакане и так далее. Все эти модели являются абстракциями, представленными вычислениями в компьютере. Для описания таких моделей применяют понятие *абстрактный тип данных*, АТД (или сложный тип данных). Строго говоря, модель движения "фигуры" в "стакане" не является типом данных, а является совокупностью операций над данными типа "фигура", использующих ограничения данных типа "стакан".

Объекты являются переменными класса. Объектно-ориентированное программирование позволяет легко создавать и использовать АТД. Объектно-ориентированное программирование использует механизм наследования. Наследование выгодно тем, что позволяет получать производные типы из уже определенных пользователем типов данных. Так, для создания фигур в тетрисе удобно сначала создать базовый класс Shape, на основе которого получены производные типы всех семи возможных в тетрисе фигур. В базовом классе определено поведение фигур, а в производных уточнена реализация поведения каждой конкретной фигуры.

В ООП объекты отвечают за свое поведение. Разработчик АТД должен включать в него код для описания любого поведения, которое обычно можно ожидать от соответствующих объектов. То, что объект сам отвечает за свое поведение, значительно упрощает задачу программирования для пользователя этого объекта.

Если мы хотим нарисовать на экране фигуру, нам надо знать где будет находиться ее центр и как ее рисовать. Если отдельная фигура прекрасно понимает, как себя нарисовать, программист при использовании такой фигуры должен лишь передать объекту сообщение "нарисоваться".

Язык MQL5 является C++ подобным, и в нем также реализован механизм инкапсуляции для реализации АТД. Инкапсуляция сочетает в себе, с одной стороны, внутренние детали реализации конкретного типа и, с другой, доступные извне функции, которые могут действовать на объекты

© 2000-2025, MetaQuotes Ltd.

этого типа. Детали реализации могут быть недоступны для программы, которая использует данный тип.

К понятию ООП имеет отношение целый набор концепций, включая нижеследующие:

- Моделирование действий из реального мира
- Наличие типов данных, определяемых пользователем
- Сокрытие деталей реализации
- Возможность многократного использования кода благодаря наследованию
- Интерпретация вызовов функций на этапе исполнения

Некоторые из этих понятий довольно расплывчаты, некоторые – абстрактны, другие носят общий характер.

© 2000-2025, MetaQuotes Ltd.

Инкапсуляция и расширяемость типов

ООП - это сбалансированный подход к написанию программного обеспечения. Данные и поведение упакованы вместе. Такая инкапсуляция создает определяемые пользователем типы данных, расширяющие собственные типы данных языка и взаимодействующие с ними. Расширяемость типов - это возможность добавлять к языку определяемые пользователем типы данных,которые также легко использовать, как и основные типы.

Абстрактный тип данных, например, строка, является описанием идеального, всем известного поведения типа. Пользователь строки знает, что операции, такие как конкатенация или печать, имеют определенное поведение. Операции конкатенации и печати называются методами.

Конкретная реализация АТД может иметь ограничения; например, строки могут быть ограничены по длине. Эти ограничения влияют на открытое всем поведение. В то же время, внутренние или закрытые детали реализации не влияют прямо на то, как пользователь видит объект. Например, строка часто реализуется как массив; при этом внутренний базовый адрес элементов этого массива и его имя не существенны для пользователя.

Инкапсуляция - это способность скрывать внутренние детали при предоставлении открытого интерфейса к определяемому пользователем типу. В MQL5, как и в C++, для обеспечения инкапсуляции используются определения класса и структуры (class и struct) в сочетании с ключевыми словами доступа private (закрытый), protected (защищенный) и public (открытый).

Ключевое слово public показывает, что доступ к членам, которые стоят за ним, является открытым безо всяких ограничений. Без этого ключевого слова члены класса по умолчанию закрыты. Закрытые члены доступны только функциям-членам только своего класса.

Защищенные члены класса доступны функциям-членам не только своего класса, но и классов-наследников. Открытые члены доступны любой функции внутри области видимости объявления класса. Закрытость позволяет спрятать часть реализации класса, предотвращая тем самым непредвиденные изменения структуры данных. Ограничение доступа или сокрытие данных является особенностью объектно-ориентированного программирования.

Обычно стараются защищать члены класса и объявлять их с модификатором protected, установку и чтение значений этих членов осуществляется с помощью специальных так называемых set- и get-методов, которые определяются с модификатором доступа public.

Пример:

```
class CPerson
  {
protected:
   string            m_name;                    // имя
public:
   void              SetName(string n){m_name=n;}// устанавливает имя
   string            GetName(){return (m_name);} // возвращает имя
  };
```

Такой подход дает несколько преимуществ. Во-первых, по имени функции можно понять что она делает - устанавливает или получает значение члена класса. Во-вторых, возможно в будущем нам понадобится изменить тип переменной m_name в самом классе CPerson или в каком-либо из производных от него классов.

© 2000-2025, MetaQuotes Ltd.

В таком случае нам достаточно будет изменить реализацию функций SetName() и GetName(), сами же объекты класса CPerson можно будет использовать в программе без каких-либо изменений в коде, так как пользователь не будет даже знать,что тип данных m_name изменился.

Пример:

```
struct Name
  {
   string            first_name;                  // имя
   string            last_name;                   // фамилия
  };

class CPerson
  {
protected:
   Name              m_name;                      // имя
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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

Типы данных

© 2000-2025, MetaQuotes Ltd.

Наследование

Особенностью ООП является поощрение повторного использования кода при помощи механизма наследования. Новый класс производится от существующего, называемого базовым классом. Производный класс использует члены базового класса, но может также изменять и дополнять их.

Многие типы представляют собой вариации на темы существующих. Часто бывает утомительно разрабатывать новый код для каждого из них. Кроме того, новый код - новые ошибки. Производный класс наследует описание базового класса, делая ненужными повторную разработку и тестирование кода. Отношения наследования иерархичны.

Иерархия - это метод, позволяющий копировать элементы во всем их многообразии и сложности. Она вводит классификацию объектов. Например, в периодической системе элементов есть газы. Они обладают свойствами, присущими всем элементам системы.

Инертные газы - следующий важный подкласс. Иерархия заключается в том, что инертный газ, например, аргон - это газ, а газ, в свою очередь является элементом системы. Такая иерархия позволяет легко толковать поведение инертных газов. Мы знаем, что их атомы содержат протоны и электроны, что верно и для прочих элементов.

Мы знаем, что они пребывают в газообразном состоянии при комнатной температуре, как все газы. Мы знаем, что ни один газ из подкласса инертных газов не вступает в обычную химическую реакцию ни с одним из элементов, и это свойство всех инертных газов.

Рассмотрим наследование на примере геометрических фигур. Для описания всего многообразия простых фигур (круг, треугольник, прямоугольник, квадрат и так далее) лучше всего создать базовый класс (АТД), который является прародителем всех производных классов.

Создадим базовый класс CShape, в котором есть только самые общие члены, описывающие фигуру. Эти члены описывают свойства, присущие любой фигуре - тип фигуры и координаты основной точки привязки.

Пример:

```
//--- Базовый класс Фигура
class CShape
  {
protected:
   int       m_type;                    // тип фигуры
   int       m_xpos;                    // X - координата точки привязки
   int       m_ypos;                    // Y - координата точки привязки
public:
             CShape(){m_type=0; m_xpos=0; m_ypos=0;} // конструктор
   void      SetXPos(int x){m_xpos=x;} // установим X
   void      SetYPos(int y){m_ypos=y;} // установим Y
  };
```

Далее создадим от базового класса производные классы, в которых добавим необходимые поля, уточняющие каждый конкретный класс. Для фигуры Circle (круг) необходимо добавить член, который содержит значение радиуса. Фигура Quadrate (квадрат) характеризуется значением стороны квадрата. Поэтому производные классы, унаследованнные от базового класса CShape, будут объявлены таким образом:

© 2000-2025, MetaQuotes Ltd.

```
//--- производный класс Круг
class CCircle : public CShape           // после двоеточия указывается базовый класс,
  {                                     // от которого производится наследование
private:
   int             m_radius;            // радиус круга

public:
                   CCircle(){m_type=1;} // конструктор, тип равен 1
  };
```

Для квадрата объявление класса выглядит аналогично:

```
//--- производный класс Квадрат
class CSquare : public CShape           // после двоеточия указывается базовый класс,
  {                                     // от которого производится наследование
private:
   int             m_square_side;       // сторона квадрата

public:
                   CSquare(){m_type=2;} // конструктор, тип равен 2
  };
```

Необходимо отметить, что при создании объекта сначала вызывается конструктор базового класса, затем конструктор производного класса. При уничтожении объекта сначала вызывается деструктор производного класса, а затем деструктор базового класса.

Таким образом, объявив в базовом классе наиболее общие члены, в производных классах мы можем добавлять дополнительные члены, которые уточняют конкретный класс. Наследование позволяет создавать мощные библиотеки кода, которые можно использовать многократно и повторно.

Синтаксис создания производного класса от уже существующего выглядит следующим образом:

```
class имя_класса :
          (public | protected | private) opt  имя_базового_класса
  {
    объявления членов
  };
```

Одним из аспектов производного класса является видимость (открытость) его членов-наследников. Ключевые слова public, protected и private используются для указания того, насколько члены базового класса будут доступны из производного. Использование в заголовке производного класса ключевого класса public, следующего за двоеточием, означает, что защищенные и открытые (protected и public) члены базового класса CShape должны наследоваться как защищенные и открытые члены производного класса CCircle.

Закрытые члены базового класса недоступны производному классу. Открытое наследование означает также, что производные классы (CCircle и CSquare) являются CShape. То есть, квадрат (CSquare) является фигурой (CShape), но фигура не обязательно должна быть квадратом.

Производный класс является модификацией базового класса; он наследует защищенные и открытые члены базового класса. Не могут только наследоваться конструкторы и деструкторы

© 2000-2025, MetaQuotes Ltd.

базового класса. Часто в производный класс добавляются новые члены в дополнение к членам базового класса.

Производный класс может включать реализацию функций-членов, отличную от базового класса. Это не имеет ничего общего с перегрузкой, когда смысл одного и того же имени функции может быть различным для разных сигнатур.

При защищенном наследовании открытые и защищенные члены базового класса становятся защищенными членами производного класса. При закрытом наследовании открытые и защищенные члены базового класса становятся закрытыми членами производного класса.

При защищенном и закрытом наследованиях не справедливо отношение, что объект производного класса является объектом базового класса. Защищенное и закрытое наследование встречаются редко и каждое из них нужно использовать с большой осторожностью.

Необходимо понимать, что тип наследования (public, protected или private) никак не влияет на способы **доступа к членам базовых классов в иерархии наследования из потомка (наследника)**. При любом типе наследования из производных классов будут доступны только члены базового класса, объявленные со спецификаторами доступа public и protected. Рассмотрим вышесказанное на примере:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| Класс-пример с несколькими типами доступа                        |
//+------------------------------------------------------------------+
class CBaseClass
  {
private:              //--- закрытый член недоступен из потомков
   int                m_member;
protected:            //--- защищенный метод доступен из базового класса и его потомков
   int                Member(){return(m_member);}
public:               //--- конструктор класса доступен всем
                      CBaseClass(){m_member=5;return;};
private:              //--- закрытый метод для присвоения значения члену m_member
   void               Member(int value) { m_member=value;};

  };
//+------------------------------------------------------------------+
//| Производный класс с ошибками                                     |
//+------------------------------------------------------------------+
class CDerived: public CBaseClass // public наследование можно не указывать, оно по ум
  {
public:
   void Func() // определим в потомке функцию с обращениями к членам базового класса
     {
      //--- попытка модификации закрытого члена базового класса
      m_member=0;         // ошибка, закрытый член базового класса никому не доступен
      Member(0);          // ошибка, закрытый метод базового класса не доступен в потом
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- чтение члена базового класса
      Print(m_member);    // ошибка, закрытый член базового класса никому не доступен
      Print(Member());    // нет ошибки, защищенный метод доступен из базового класса и
     }
  };
```

В приведенном примере класс CBaseClass имеет только один публичный метод - конструктор. Конструкторы вызываются автоматически при создании объекта класса. Поэтому извне никак нельзя обратиться ни к закрытому члену m_member, ни к защищенному методу Member(). Но при этом при открытом (public) наследовании метод Member() базового класса будет доступен из потомков.

При защищенном (protected) наследовании все члены базового класса с открытым и защищенным доступом становятся защищенными. Это означает, что если открытые члены-данные и методы базового класса были доступны извне, то при защищенном наследовании теперь они доступны только из классов самого потомка и его последующих производных классах.

```
//+------------------------------------------------------------------+
//| Класс-пример с несколькими типами доступа                        |
//+------------------------------------------------------------------+
class CBaseMathClass
  {
private:              //--- закрытый член недоступен из потомков
   double             m_Pi;
public:               //--- получение и установка значения для m_Pi
   void               SetPI(double v){m_Pi=v;return;};
   double             GetPI(){return m_Pi;};
public:               // конструктор класса доступен всем
                      CBaseMathClass() {SetPI(3.14);  PrintFormat("%s",__FUNCTION__);};
  };
//+------------------------------------------------------------------+
//| Производный класс, в котором нельзя уже изменить m_Pi            |
//+------------------------------------------------------------------+
class CProtectedChildClass: protected CBaseMathClass // защищенное наследование
  {
private:
   double             m_radius;
public:               //--- открытые методы в потомке
   void               SetRadius(double r){m_radius=r; return;};
   double             GetCircleLength(){return GetPI()*m_radius;};
  };
//+------------------------------------------------------------------+
//| Функция запуска скрипта                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- при создании потомка конструктор базового класса вызовется автоматически
   CProtectedChildClass pt;
//--- укажем радиус
   pt.SetRadius(10);
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("Length=%G",pt.GetCircleLength());
//--- если раскомментировать строку ниже, то получим ошибку на этапе компиляции,так ка
// pt.SetPI(3);

//--- а теперь объявим переменную базового класса и попробуем задать константу Pi равн
   CBaseMathClass bc;
   bc.SetPI(10);
//--- посмотрим что получилось
   PrintFormat("bc.GetPI()=%G",bc.GetPI());
  }
```

В данном пример показано, что методы SetPI() и GetPi() в базовом классе CBaseMathClass являются открытыми и доступны для вызова из любого места программы. Но в то же время для его потомка CProtectedChildClass вызовы этих методов можно делать только из методов самого класса CProtectedChildClass или его потомков.

При закрытом (private) наследовании все члены базового класса с доступом public и protected становятся закрытыми, и при дальнейшем наследовании обращение к ним становится невозможным.

В MQL5 нет множественного наследования.

Смотри также

Структуры и классы

© 2000-2025, MetaQuotes Ltd.

Полиморфизм

Полиморфизм - это возможность для объектов разных классов, связанных с помощью наследования, реагировать различным способом при обращении к одной и той же функции-элементу. Это помогает создавать универсальные механизмы, описывающие поведение не только базового класса, но и классов-потомков.

Продолжим разработку базового класса CShape, в котором определим функцию-член GetArea(), предназначенную для расчета площади фигуры. Во всех классах-потомках, произведенных наследованием от базового класса, мы переопределим эту функцию в соответствие с правилами расчета площади конкретной фигуры.

Для квадрата (класс CSquare) площадь вычисляется через стороны, для круга (класс CCircle) площадь выражается через радиус и так далее. Мы можем создать массив для хранения объектов типа CShape, в котором можно будет хранить как объект базового класса, так и всех его потомков. В дальнейшем мы можем вызывать одну и ту же функцию для любого элемента данного массива.

Пример:

```
//--- Базовый класс
class CShape
  {
protected:
   int            m_type;                 // тип фигуры
   int            m_xpos;                 // X - координата точки привязки
   int            m_ypos;                 // Y - координата точки привязки
public:
   void           CShape(){m_type=0;};    // конструктор, тип равен нулю
   int            GetType(){return(m_type);};// возвращает тип фигуры
virtual
   double         GetArea(){return (0); }// возвращает площадь фигуры
  };
```

Теперь все производные классы имеют функцию-член getArea(), которая возвращает нулевое значение. Реализация этой функции в каждом потомке будет отличаться.

```
//--- производный класс Круг
class CCircle : public CShape            // после двоеточия указывается базовый класс,
  {                                      // от которого производится наследование
private:
   double         m_radius;               // радиус круга

public:
   void           CCircle(){m_type=1;};  // конструктор, тип равен 1
   void           SetRadius(double r){m_radius=r;};
   virtual double GetArea(){return (3.14*m_radius*m_radius);}// площадь круга
  };
```

Для квадрата объявление класса выглядит аналогично:

```
//--- производный класс Квадрат
```

© 2000-2025, MetaQuotes Ltd.

```
class CSquare : public CShape            // после двоеточия указывается базовый класс,
  {                                      // от которого производится наследование
private:
   double          m_square_side;        // сторона квадрата

public:
   void            CSquare(){m_type=2;}; // конструктор, тип равен 2
   void            SetSide(double s){m_square_side=s;};
   virtual double  GetArea(){return (m_square_side*m_square_side);}//площадь квадрата
  };
```

Так как для вычисления площади квадрата и круга требуются соответствующие значения членов m_radius и m_square_side, то в объявлении соответствующего класса мы добавили функции SetRadius() и SetSide().

Предполагается, что в программе у нас используются объекты разного типа (CCircle и CSquare), но унаследованные от одного базового типа CShape. Полиморфизм позволяет нам создать массив объектов базового типа CShape, но при объявлении этого массива сами объекты еще неизвестны и их тип неопределен.

Решение о том, объект какого типа будет содержаться в каждом элементе массива, будет приниматься непосредственно при выполнении программы. Это подразумевает динамическое создание объектов соответствующих классов, и, следовательно, необходимость вместо самих объектов использовать указатели объектов.

Для динамического создания объектов используется оператор new, каждый такой объект нужно самостоятельно и явно удалять оператором delete. Поэтому мы объявим массив указателей типа CShape и для каждого его элемента создадим объект нужного типа (**new Имя_класса**), как это показано в примере скрипта:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим массив указателей объектов базового типа
   CShape *shapes[5];   // массив указателей на объекты CShape

//--- здесь заполняем массив производными объектами
//--- объявим указатель на объект типа CCircle
   CCircle *circle=new CCircle();
//--- задаем свойства объекта по указателю circle
   circle.SetRadius(2.5);
//--- поместим в shapes[0] значение указателя
   shapes[0]=circle;

//--- создаем еще один объект CCircle и запишем его указатель в shapes[1]
   circle=new CCircle();
   shapes[1]=circle;
   circle.SetRadius(5);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- тут мы намеренно "забыли" задать значение для shapes[2]
//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;

//--- для неиспользуемого элемента установим значение NULL
   shapes[2]=NULL;

//--- создаем объект CSquare и запишем его указатель в shapes[3]
   CSquare *square=new CSquare();
   square.SetSide(5);
   shapes[3]=square;

//--- создаем объект CSquare и запишем его указатель в shapes[4]
   square=new CSquare();
   square.SetSide(10);
   shapes[4]=square;

//--- массив указателей есть, получим его размер
   int total=ArraySize(shapes);
//--- пройдем в цикле по всем указателям в массиве
   for(int i=0; i<5;i++)
     {
      //--- если по указанному индексу указатель является валидным
      if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
         //--- выведем в лог тип и площадь фигуры
         PrintFormat("Объект типа %d имеет площадь %G",
               shapes[i].GetType(),
               shapes[i].GetArea());
        }
      //--- если указатель имеет тип POINTER_INVALID
      else
        {
         //--- сообщим об ошибке
         PrintFormat("Объект shapes[%d] не инициализирован! Его указатель %s",
                     i,EnumToString(CheckPointer(shapes[i])));
        }
     }

//--- мы должны самостоятельно уничтожить все созданные динамические объекты
   for(int i=0;i<total;i++)
     {
      //--- удалять можно только объекты, чей указатель имеет тип POINTER_DYNAMIC
      if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
         //--- сообщим об удалении
         PrintFormat("Удаляем shapes[%d]",i);
         //--- уничтожим объект по его указателю
```

© 2000-2025, MetaQuotes Ltd.

```
         delete shapes[i];
        }
     }
  }
```

Обратите внимание, что при уничтожении объекта оператором delete необходимо проверить тип его указателя. Удалять с помощью delete можно только объекты, имеющие указатель POINTER_DYNAMIC, для указателей другого типа будет получена ошибка.

Кроме переопределения функции при наследовании, полиморфизм включает в себя также и реализацию одной и той же функции с разным набором параметров в пределах одного класса. Это означает, что в классе может быть определено несколько функций с одним и тем же именем, но с разным типом и/или набором параметров. В этом случае полиморфизм реализуется через перегрузку функций.

Смотри также

Стандартная библиотека

© 2000-2025, MetaQuotes Ltd.

Перегрузка

В пределах одного класса можно определить два или более методов, которые совместно используют одно и тоже имя, но имеют разное количество параметров. Когда это имеет место, методы называются *перегруженными*, а о процессе говорят как о *перегрузке метода*. Перегрузка метода – один из способов, с помощью которых реализуется полиморфизм. Перегрузка методов в классах производится по тем же правилам, что и перегрузка функций.

Если для вызываемой функции нет точного соответствия, то компилятор производит поиск подходящей функции по трем уровням последовательно:

1. поиск среди методов класса;
2. поиск среди методов базовых классов, последовательно от ближайшего предка до самого первого;
3. поиск среди остальных функций.

Если точного соответствия ни на одном уровне не найдено, но найдено несколько подходящих функций на разных уровнях, то используется функция, найденная на наименьшем уровне. В пределах одного уровня не может быть более одной подходящей функции.

Смотри также

Перегрузка функций

© 2000-2025, MetaQuotes Ltd.

Виртуальные функции

Ключевое слово virtual служит спецификатором функции, который обеспечивает механизм для динамического выбора на этапе выполнения подходящей функции-члена среди функций базового и производного классов, структуры не могут иметь виртуальных функций. Оно может применяться для изменения объявлений только функций-членов.

Виртуальная функция, как и обычная функция, должна иметь исполняемое тело. При вызове семантика ее точно такая же, как и у остальных функций.

Виртуальная функция может замещаться в производном классе. Выбор того, какое определение функции вызвать для виртуальной функции, происходит динамически (на этапе выполнения). Типичный случай - когда базовый класс содержит виртуальную функцию, а производные классы имеют свои версии этой функции.

Указатель на базовый класс может указывать либо на объект базового класса, либо на объект производного класса. Выбор вызываемой функции-члена будет произведен на этапе выполнения и будет зависеть от типа объекта, а не от типа указателя. При отсутствии члена производного типа по умолчанию используется виртуальная функция базового класса.

Деструкторы всегда являются виртуальными, независимо от того, объявлены они с ключевым слово virtual или нет.

Рассмотрим использование виртуальных функций на примере программы MT5_Tetris.mq5. Во включаемом файле MT5_TetrisShape.mqh определен базовый класс CTetrisShape с виртуальной функцией Draw (рисовать).

```
//+------------------------------------------------------------------+
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;    }
```

© 2000-2025, 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[]);
  };
```

Далее для каждого производного класса эта функция реализована в соответствии с особенностями класса-потомка. Например, первая фигура CTetrisShape1 имеет свою реализацию функции Draw():

```
class CTetrisShape1 : public CTetrisShape
  {
public:
   //--- отрисовка фигуры
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      if(m_turn==0 || m_turn==2)
        {
         //--- горизонтальная палка
         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
        {
         //--- вертикальная палка
         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);
           }
        }
     }
  }
```

Фигура квадрат описана классом CTetrisShape6 и имеет собственную реализацию метода Draw():

```
class CTetrisShape6 : public CTetrisShape
  {
public:
   //--- отрисовка фигуры
   virtual void      Draw()
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      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);
        }
     }
  };
```

В зависимости от того, объект какого класса создан, вызывается виртуальная функция того или иного производного класса.

```
void CTetrisField::NewShape()
  {
//--- случайным образом создаём одну из 7 возможных фигур
   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;
     }
//--- отрисовываем
   m_shape.Draw();
//---
  }
```

Модификатор override

Модификатор override означает, что объявляемая функция обязательно должна переопределить метод родительского класса. Использование этого модификатора позволяет избежать ошибок при переопределении, таких как случайное изменение сигнатуры метода. Например, в базовом классе определен метод func, принимающий в качестве аргумента переменную типа int:

```
class CFoo
  {
   void virtual func(int x) const { }
```

© 2000-2025, MetaQuotes Ltd.

```
  };
```

Далее метод переопределяется в наследуемом классе:

```
class CBar : public CFoo
  {
   void func(short x) { }
  };
```

Но по ошибке тип аргумента изменяется с int на short. Фактически, в этом случае уже происходит не переопределение, а перегрузка метода. Действуя в соответствии с алгоритмом определения перегруженной функции, в определенных ситуациях компилятор может выбрать метод, определенный в базовом классе, вместо переопределенного метода.

Чтобы избежать подобных ошибок, к переопределяемому методу следует явно добавлять модификатор override.

```
class CBar : public CFoo
  {
   void func(short x) override { }
  };
```

Если при переопределении будет изменена сигнатура метода, компилятор не сможет найти в родительском классе метод с точной такой же сигнатурой и выдаст ошибку компиляции:

```
'CBar::func' method is declared with 'override' specifier but does not override any ba
```

Модификатор final

Модификатор final действует наоборот — он запрещает переопределение метода в классах-наследниках. Если реализация метода самодостаточна и полностью завершена, объявите его с модификатором final, чтобы он гарантированно не был изменен в последующем.

```
class CFoo
  {
   void virtual func(int x) final { }
  };

class CBar : public CFoo
  {
   void func(int) { }
  };
```

При попытке переопределения метода с модификатором final, как показано в примере выше, компилятор выдаст ошибку:

```
'CFoo::func' method declared as 'final' cannot be overridden by 'CBar::func'
see declaration of 'CFoo::func'
```

Смотри также

Стандартная библиотека

© 2000-2025, MetaQuotes Ltd.

Статические члены класса/структуры

Статические члены

Члены класса могут быть объявлены с использованием модификатора класса памяти static. Такие члены данных разделяются всеми экземплярами данного класса и хранятся в одном месте. Нестатические члены данных создаются для каждой переменной-объекта класса.

Отсутствие возможности объявлять статически члены класса привело бы к необходимости объявлять эти данные на глобальном уровне программы. Это разорвало бы отношения между данными и их классом, а также не согласуется с основной парадигмой ООП - объединение в классе данных и методов для их обработки. Статический член позволяет данным класса, которые не специфичны для отдельного экземпляра, существовать в области видимости класса.

Так как статический член класса не зависит от конкретного экземпляра, то обращение к нему выглядит следующим образом:

```
class_name::variable
```

где *class_name* - это имя класса, а *variable* означает имя члена класса.

Как видите, для обращения к статическому члену класса используется оператор разрешения контекста ::. При обращении к статическому члену внутри методов класса оператор контекста необязателен.

Статический член класса требуется явно инициализировать нужным значением, для этого он должен быть объявлен и проинициализирован на глобальном уровне. Порядок инициализации статических членов будет соответствовать порядку их объявления на глобальном уровне в исходном коде.

Например, у нас есть класс *CParser*, предназначенный для синтаксического разбора текстов, и нам необходимо считать общее количество обработанных слов и символов. Достаточно объявить нужные члены класса статическими и инициализировать их на глобальном уровне. Тогда все экземпляры класса при работе будут использовать общие счетчики слов и символов.

```
//+------------------------------------------------------------------+
//| Класс "Анализатор текстов"                                       |
//+------------------------------------------------------------------+
class CParser
  {
public:
   static int        s_words;
   static int        s_symbols;
   //--- конструктор и деструктор
                     CParser(void);
                    ~CParser(void){};
  };
...
//--- инициализация статических членов класса Parser на глобальном уровне
int CParser::s_words=0;
int CParser::s_symbols=0;
```

© 2000-2025, MetaQuotes Ltd.

Статический член класса можно объявить с ключевым словом *const*. Такие статические константы должны быть инициализированы на глобальном уровне с ключевым словом *const:*

```
//+------------------------------------------------------------------+
//| Класс "Стек" для хранения обрабатываемых данных                  |
//+------------------------------------------------------------------+
class CStack
  {
public:
                     CStack(void);
                    ~CStack(void){};
...
private:
   static const int  s_max_length; // максимальная емкость стека
  };

//--- инициализация статической константы класса CStack
const int CStack::s_max_length=1000;
```

Указатель this

Ключевое слово this обозначает объявленный неявно указатель на себя - на конкретный экземпляр класса, в контексте которого выполняется метод. Он может использоваться только в нестатических методах класса. Указатель this является неявным нестатическим членом любого класса.

В статических функциях можно обращаться только к статическим членам/методам класса.

Статические методы

В MQL5 разрешается использовать функции-члены типа static. Модификатор *static* должен идти перед возвращаемым типом функции в объявлении внутри класса.

```
class CStack
  {
public:
   //--- конструктор и деструктор
                     CStack(void){};
                    ~CStack(void){};
   //--- максимальная емкость стека
   static int        Capacity();
private:
   int               m_length;     // количество элементов в стеке
   static const int  s_max_length; // максимальная емкость стека
  };
//+------------------------------------------------------------------+
//| Возвращает максимальную вместимость стека                        |
//+------------------------------------------------------------------+
int CStack::Capacity(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   return(s_max_length);
  };
//--- инициализация статической константы класса CStack
const int CStack::s_max_length=1000;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим переменную типа CStack
   CStack stack;
//--- вызываем статический метод объекта
   Print("CStack.s_max_length=",stack.Capacity());
//--- а можно и так вызвать, т.к. метод статический и не требует наличия объекта
   Print("CStack.s_max_length=",CStack::Capacity());
  }
```

Метод с модификатором const называется постоянным и не может модифицировать неявные члены своего класса. Объявление постоянных функций класса и постоянных параметров называется *контролем постоянства* (const-correctness). Благодаря такому контролю можно быть уверенным, что компилятор проследит за неизменностью значений объектов и выдаст ошибку еще на стадии компиляции в случае нарушения.

Модификатор const ставится после списка аргументов внутри объявления класса. Определение вне класса также должно включать модификатор *const*:

```
//+------------------------------------------------------------------+
//| Класс "Прямоугольник"                                            |
//+------------------------------------------------------------------+
class CRectangle
  {
private:
   double            m_width;      // ширина
   double            m_height;     // высота
public:
   //--- конструкторы и деструктор
                     CRectangle(void):m_width(0),m_height(0){};
                     CRectangle(const double w,const double h):m_width(w),m_height(h)
                    ~CRectangle(void){};
   //--- вычисление площади
   double            Square(void) const;
   static double     Square(const double w,const double h);// { return(w*h); }
  };
//+------------------------------------------------------------------+
//| Возвращает площадь объекта "Прямоугольник"                       |
//+------------------------------------------------------------------+
double CRectangle::Square(void) const
  {
   return(Square(m_width,m_height));
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Возвращает произведение двух переменных                          |
//+------------------------------------------------------------------+
static double CRectangle::Square(const double w,const double h)
  {
   return(w*h);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим прямоугольник rect со сторонами 5 и 6
   CRectangle rect(5,6);
//--- выведем площадь прямоугольника константным методом
   PrintFormat("rect.Square()=%.2f",rect.Square());
//--- выведем произведние чисел с помощью статического метода класса CRectangle
   PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
  }
```

Дополнительным аргументом в пользу использования контроля целостности служит то, что компилятор в этом случае производит специальную оптимизацию, например, располагает постоянный объект в памяти только для чтения.

Статическая функция не может быть определена с модификатором const, так как данный модификатор гарантирует неизменность членов экземпляра при вызове такой функции. Но, как уже говорилось выше, статическая функция по определению не может обращаться к нестатическим членам класса.

Смотри также

Статические переменные, Переменные, Ссылки. Модификатор & и ключевое слово this

© 2000-2025, MetaQuotes Ltd.

Шаблоны функций

Перегруженные функции обычно используются для выполнения похожих операций над различными типами данных. Простой пример такой функции в MQL5 - ArraySize(), которая возвращает размер массива любого типа. На самом деле эта системная функция является перегруженной, и вся реализация такой перегрузки спрятана от разработчика программ на MQL5:

```
int  ArraySize(
   void&  array[]      // проверяемый массив
   );
```

То есть на самом деле компилятор языка MQL5 подставляет для каждого вызова этой функции нужную реализацию, например, для массивов целого типа примерно так:

```
int  ArraySize(
   int&  array[]      // массив с элементами типа int
   );
```

А для массива типа MqlRates для работы с котировками в формате исторических данных функцию ArraySize() можно представить таким образом:

```
int  ArraySize(
   MqlRates&  array[] // массив, заполненный значениями типа MqlRates
   );
```

Таким образом, очень удобно использовать одну и ту же функцию для работы с разными типами, но необходимо самостоятельно провести всю предварительную работу, а именно - сделать перегрузку нужной функции для всех типов данных, с которыми она должна будет корректно работать.

Есть более удобное решение - если для каждого типа данных должны выполняться идентичные операции, то более компактным и удобным решением является использование шаблонов функций. При этом программисту нужно написать всего одно описание шаблона функции. При таком описании шаблона в качестве параметра достаточно указать не конкретный тип данных, с которыми должна работать функция, а некий формальный параметр. Основываясь на типах аргументов, использованных при вызове этой функции, компилятор будет автоматически генерировать разные функции для соответствующей обработки каждого типа.

Определение шаблона функции начинается с ключевого слова template, после которого в угловых скобках идет список формальных параметров. Каждый формальный параметр предваряется ключевым словом typename. Формальные типы параметров - встроенные типы или типы, определенные пользователем. Они используются:

- для задания типов аргументов функции,
- для задания типов возвращаемого значения функции,
- для объявления переменных внутри тела описания функции

Количество параметров в шаблоне не может быть больше восьми. Каждый формальный параметр в определении шаблона должен хотя бы один раз появиться в списке параметров функции. Каждое имя формального параметра должно быть уникальным.

© 2000-2025, MetaQuotes Ltd.

Вот пример шаблона функции для поиска максимального значения в массиве любого числового типа (целые и вещественные числа):

```
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);
  }
```

Данный шаблон описывает функцию, которая находит максимальное значение в переданном массиве и возвращает его в качестве результата. Напомним, что встроенная в MQL5 функция ArrayMaximum() возвращает только индекс найденного максимального значения, по которому пользователь в дальнейшем может получить уже и само значение. Например, так:

```
//--- создадим массив
   double array[];
   int size=50;
   ArrayResize(array,size);
//---  заполним случайными значениями
   for(int i=0;i<size;i++)
      array[i]=MathRand();

//--- найдем позицию максимального элемента в массиве
   int max_position=ArrayMaximum(array);
//--- теперь получим само максимальное значение в массиве
   double max=array[max_position];
//--- вывод найденного значения
   Print("Max value = ",max);
```

Таким образом, нам понадобилось два действия, чтобы получить максимальное значение в массиве. С помощью шаблона функции ArrayMax() можно получить результат нужного типа, просто передав в неё массив соответствующего типа. То есть вместо двух последних строчек

```
//--- найдем позицию максимального элемента в массиве
   int max_position=ArrayMaximum(array);
//--- теперь получим само максимальное значение в массиве
   double max=array[max_position];
```

теперь мы можем использовать одну строку, возвращающую сразу результат того же типа, что и переданный массив:

```
//--- найдем максимальное значение
   double max=ArrayMax(array);
```

© 2000-2025, MetaQuotes Ltd.

При этом тип результата, возвращенный функцией ArrayMax(), будет автоматически соответствовать типу массива.

Для создания универсальных способов работы с различными типами данных необходимо использовать ключевое слово typename для получения типа аргумента в виде строки. Покажем это на примере функции, которая возвращает в виде строки тип данных:

```
#include <Trade\Trade.mqh>
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CTrade trade;
   double d_value=M_PI;
   int i_value=INT_MAX;
   Print("d_value: type=",GetTypeName(d_value), ",   value=", d_value);
   Print("i_value: type=",GetTypeName(i_value), ",   value=", i_value);
   Print("trade: type=",GetTypeName(trade));
//---
  }
//+------------------------------------------------------------------+
//| Возвращает в строковом виде тип                                  |
//+------------------------------------------------------------------+
template<typename T>
string GetTypeName(const T &t)
  {
//--- вернем тип в виде строки
   return(typename(T));
//---
  }
```

Шаблоны функций можно также использовать и для методов класса, например:

```
class CFile
  {
   ...
public:
   ...
   template<typename T>
   uint WriteStruct(T &data);
  };

template<typename T>
uint CFile::WriteStruct(T &data)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ...
   return(FileWriteStruct(m_handle,data));
  }
```

Шаблоны функций нельзя объявлять с ключевыми словами export, virtual и #import.

Перегрузка шаблонных функций

В некоторых случаях может понадобиться перегрузка шаблонной функции. Например, у нас есть шаблонная функция, которая записывает в первый параметр значение второго параметра с помощью явного приведения типов. В языке MQL5 запрещено приведение типа string к типу bool, мы можем сделать это сами - для этого создадим перегрузку шаблонной функции. Например:

```
//+------------------------------------------------------------------+
//| Шаблонная функция                                                |
//+------------------------------------------------------------------+
template<typename T1,typename T2>
string Assign(T1 &var1,T2 var2)
  {
   var1=(T1)var2;
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Специальная перегрузка для случая 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"));
  }
```

В результате выполнения данного кода мы увидим, что для пары int+string была использована шаблонная функция Assign(), а при втором вызове для пары bool+string уже использовалась перегруженная версия.

```
string Assign<int,string>(int&,string)
string Assign(bool&,string)
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

Перегрузка

© 2000-2025, MetaQuotes Ltd.

Чем хороши шаблоны

Шаблоны функций используются в тех случаях, когда необходимо производить одинаковые операции с данными разного типа, например – поиск максимального элемента в массиве. Главное преимущество использования шаблонов заключается в том, что программисту нет необходимости писать отдельную перегрузку для каждого типа. То есть вместо объявления множества перегрузок для каждого типа

```
double ArrayMax(double array[])
  {
   ...
  }
int ArrayMax(int array[])
  {
   ...
  }
uint ArrayMax(uint array[])
  {
   ...
  }
long ArrayMax(long array[])
  {
   ...
  }
datetime ArrayMax(datetime array[])
  {
   ...
  }
```

достаточно написать одну шаблонную функцию

```
template<typename T>
T ArrayMax(T array[])
  {
   if(ArraySize()==0)
      return(0);
   uint max_index=ArrayMaximum(array);
   return(array[max_index]);
  }
```

и затем использовать её в своем коде:

```
double high[];
datetime time[];
....
double max_high=ArrayMax(high);
datetime lasttime=ArrayMax(time);
```

Здесь формальный параметр *T*, который задает тип используемых данных, при компиляции заменяется на реально используемый тип, то есть компилятор автоматически генерирует

© 2000-2025, MetaQuotes Ltd.

отдельную функцию под каждый тип - double, datetime и так далее. Точно также в языке MQL5 можно создавать шаблоны классов, используя все преимущества такого подхода.

Шаблоны классов

Шаблон класса объявляется с помощью ключевого слов template, за которым идут угловые скобки <>, в которых перечисляется список формальных параметров с ключевым словом typename. Такая запись указывает компилятору, что перед ним обобщенный класс с формальным параметром *T*, задающим реальный тип переменной при реализации класса. Например, создадим класс-вектор для хранения массива с элементами типа *T*:

```
#define TOSTR(x) #x+" "   // макрос для вывода имени объекта
//+------------------------------------------------------------------+
//| Класс-вектор для хранения элементов типа T                       |
//+------------------------------------------------------------------+
template <typename T>
class TArray
  {
protected:
   T                 m_array[];
public:
   //--- конструктор по умолчанию создает массив на 10 элементов
   void TArray(void){ArrayResize(m_array,10);}
   //--- конструктор для создания вектора с заданным размером массива
   void TArray(int size){ArrayResize(m_array,size);}
   //--- возвращает тип и количество данных, которые хранятся в объекте типа TArray
   string Type(void){return(typename(m_array[0])+":"+(string)ArraySize(m_array));};
  };
```

Далее, в программе создадим разными способами три объекта *TArray* для работы с разными типами

```
void OnStart()
  {
   TArray<double> double_array;    // вектор имеет размер по умолчанию 10
   TArray<int> int_array(15);      // вектор имеет размер 15
   TArray<string> *string_array;   // указатель на вектор TArray<string>
//--- создадим динамический объект
   string_array=new TArray<string>(20);
//--- выведем в Журнал имя объекта, тип данных и размер вектора
   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());
//--- удалим динамический объект перед завершением программы
   delete(string_array);
  }
```

Результат выполнения скрипта:

```
  double_array  (double:10)
  int_array  (int:15)
```

© 2000-2025, MetaQuotes Ltd.

```
  string_array  (string:20)
```

В результате было создано 3 вектора с разными типами данных: double, int и string.

Шаблоны классов хорошо подходят для разработки контейнеров - объектов, которые предназначены для инкапсуляции объектов любого типа. Объектами контейнеров являются коллекции, которые уже содержат объекты одного определенного типа. Как правило в контейнер сразу же встраивается и реализация по работе с данными, которые в нём хранятся.

Например, можно создать шаблон класса, который не позволяет обратиться к элементу за пределами массива, и таким образом избегать критической ошибки "out of range".

```
//+------------------------------------------------------------------+
//| Класс для безопасного обращения к элементу массива               |
//+------------------------------------------------------------------+
template<typename T>
class TSafeArray
  {
protected:
   T                 m_array[];
public:
   //--- конструктор по умолчанию
   void              TSafeArray(void){}
   //--- конструктор для создания массива заданного размера
   void              TSafeArray(int size){ArrayResize(m_array,size);}
   //--- размер массива
   int               Size(void){return(ArraySize(m_array));}
   //--- изменение размера массива
   int               Resize(int size,int reserve){return(ArrayResize(m_array,size,rese
   //--- освобождение массива
   void              Erase(void){ZeroMemory(m_array);}
   //--- оператор доступа к элементу массива по индексу
   T                 operator[](int index);
   //--- оператор присваивания для получения сразу всех элементов из массива
   void              operator=(const T  &array[]); // массив типа T
  };
//+------------------------------------------------------------------+
//| Операция получения элемента по индексу                           |
//+------------------------------------------------------------------+
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);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(m_array[index]);
  }
//+------------------------------------------------------------------+
//| Операция присваивания для массива                                |
//+------------------------------------------------------------------+
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   int size=ArraySize(array);
   ArrayResize(m_array,size);
//--- тип T должен поддерживать оператор копирования
   for(int i=0;i<size;i++)
      m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int copied,size=15;
   MqlRates rates[];
//--- скопируем массив котировок
   if((copied=CopyRates(_Symbol,_Period,0,size,rates))!=size)
     {
      PrintFormat("CopyRates(%s,%s,0,%d) вернула код ошибки %d",
      _Symbol,EnumToString(_Period),size,GetLastError());
      return;
     }
//--- создадим контейнер и вложим в него массив значений MqlRates
   TSafeArray<MqlRates> safe_rates;
   safe_rates=rates;
   //--- индекс в пределах массива
   int index=3;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
   //--- индекс за пределами массива
   index=size;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
  }
```

Обратите внимание, что при описании методов за пределами декларации класса также необходимо использовать объявление шаблона:

```
template<typename T>
T TSafeArray::operator[](int index)
  {
   ...
  }
```

© 2000-2025, MetaQuotes Ltd.

```
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   ...
  }
```

Шаблоны классов и функций позволяют указывать несколько формальных параметров через запятую, например, коллекция Map для хранения пар "ключ - значение":

```
template<typename Key, template Value>
class TMap
  {
   ...
  }
```

Смотри также

Шаблоны функций, Перегрузка

© 2000-2025, MetaQuotes Ltd.

Абстрактные классы и чисто виртуальные функции

Абстрактные классы предназначены для создания обобщенных сущностей, на основе которых в дальнейшем предполагается создавать более конкретные производные классы. Абстрактный класс - это класс, который может использоваться лишь в качестве базового класса для некоторого другого класса, поэтому невозможно создать объект типа абстрактного класса

Класс, содержащий хотя бы одну чисто виртуальную функцию, является абстрактным. Поэтому, классы, производные от абстрактного класса, должны реализовать все его чисто виртуальные функции, иначе они также будут абстрактными классами.

Виртуальная функция объявляется как "чистая" с помощью синтаксиса спецификатора-чистоты. Рассмотрим в качестве примера класс CAnimal, который создаётся только для того, чтобы предоставлять общие функции - сами объекты типа CAnimal имеют слишком общий характер для практического применения. Таким образом, класс CAnimal является хорошим кандидатом в абстрактный класс:

```
class CAnimal
  {
public:
                      CAnimal();      // конструктор
   virtual void       Sound() = 0;    // чисто виртуальная функция
private:
   double             m_legs_count;   // количество ног животного
  };
```

Здесь функция Sound() является чисто виртуальной, потому что она объявлена со спецификатором чисто виртуальной функции PURE (=0).

Чисто виртуальными функциями являются только такие виртуальные функции, для которых указан спецификатор чистоты PURE, а именно: (=NULL) или (=0). Пример объявления и использования абстрактного класса:

```
class CAnimal
  {
public:
   virtual void       Sound()=NULL;   // PURE method, должен быть переопределён в пото
  };
//--- потомок от абстрактного класса
class CCat : public CAnimal
 {
public:
  virtual void        Sound() { Print("Myau"); } // PURE переопределён, класс CCat не
 };

//--- примеры неправильного использования
new CAnimal;          // ошибка 'CAnimal' - компилятор выдаст ошибку "cannot instantiat
CAnimal some_animal; // ошибка 'CAnimal' - компилятор выдаст ошибку "cannot instantiat

//--- примеры правильного использования
new CCat;  // ошибки нет - класс CCat не абстрактный
```

© 2000-2025, MetaQuotes Ltd.

```
CCat cat;  // ошибки нет - класс CCat не абстрактный
```

Ограничения на использование абстрактных классов

При вызове конструктором абстрактного класса чистой виртуальной функции (прямо или косвенно) результат будет неопределённым.

```
//+------------------------------------------------------------------+
//| Абстрактный базовый класс                                        |
//+------------------------------------------------------------------+
class CAnimal
  {
public:
   //--- чисто виртуальная функция
   virtual void      Sound(void)=NULL;
   //--- функция
   void              CallSound(void) { Sound(); }
   //--- конструктор
   CAnimal()
    {
     //--- явный вызов виртуального метода
     Sound();
     //--- неявный вызов (через третью функцию)
     CallSound();
     //--- в конструкторе и/или деструкторе всегда вызываются свои функции,
     //--- несмотря на виртуальность и переопределение вызываемой функции в потомке
     //--- если вызываемая функция чисто виртуальная, то
     //--- вызов приведёт к критической ошибке выполнения: "pure virtual function cal
    }
  };
```

Однако конструкторы и деструкторы абстрактных классов могут вызывать другие функции-члены.

© 2000-2025, MetaQuotes Ltd.

Пространства имен

Пространство имен — это специально объявленная область, в пределах которой определяются различные идентификаторы: переменные, функции, классы и и т. д. Задается с помощью ключевого слова namespase:

```
namespace имя_пространства {
  // список определений функций, классов и переменных
}
```

Использование namespace позволяет разделить глобальное пространство имен на подпространства. Все идентификаторы в пределах пространства имен доступны друг другу без уточнения. Для доступа к членам пространства имен извне используется оператор :: (операция разрешения контекста).

```
namespace ProjectData
{
class DataManager
  {
public:
   void              LoadData() {}
  };
void Func(DataManager& manager) {}
}
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- работа с пространством имен ProjectData
   ProjectData::DataManager mgr;
   mgr.LoadData();
   ProjectData::Func(mgr);
  }
```

Пространства имен используются для организации кода в виде логических групп и с целью избежания конфликтов имен, которые могут возникнуть когда в программе используется несколько библиотек. В таких случаях каждую библиотеку можно объявить в своем пространстве имен, чтобы затем явно обращаться к нужным функциям и классам каждой библиотеки.

Пространство имен может быть объявлено в нескольких блоках в одном файле и в нескольких файлах. Компилятор соединит вместе все части во время предварительной обработки и полученное в результате пространство имен будет содержать все члены, объявленные во всех частях. Пусть у нас реализован класс A во включаемом файле Sample.mqh:

```
//+------------------------------------------------------------------+
//|                                                   Sample.mqh |
//+------------------------------------------------------------------+
class A
  {
public:
```

© 2000-2025, MetaQuotes Ltd.

```
                     A() {Print(__FUNCTION__);}
  };
```

Этот класс мы хотим использовать в своем проекте, но у нас уже есть класс A. Чтобы иметь возможность использовать оба класса и избежать конфликта идентификаторов, достаточно включаемый файл обернуть в пространство имен:

```
//--- объявим первый класс A
class A
  {
public:
                     A() {Print(__FUNCTION__);}
  };
//--- обернем класс A из файла "Sample.mqh" в пространство имен "Library" для избежани
namespace Library
{
#include "Sample.mqh"
}
//--- добавим еще один класс в пространство имен "Library"
namespace Library
{
class B
  {
public:
                     B() {Print(__FUNCTION__);}
  };
}
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- используем класс A из глобального пространства имен
   A a1;
//--- используем классы A и B из пространства имен "Library"
   Library::A a2;
   Library::B b;
  }
//+------------------------------------------------------------------+

/*
Результат:
   A::A
   Library::A::A
   Library::B::B
*/
```

Пространства имен могут быть вложенными. Вложенное пространство имен имеет неограниченный доступ к членам своего родительского пространства, но члены родительского пространства не имеют неограниченного доступа к вложенному пространству имен.

© 2000-2025, MetaQuotes Ltd.

```
namespace General
{
int Func();

namespace Details
{
 int Counter;
 int Refresh()  {return Func(); }
}

int GetBars()   {return(iBars(Symbol(), Period()));};
int Size(int i) {return Details::Counter;}
}
```

Глобальное пространство имен

Если идентификатор не объявлен явно в пространстве имен, он неявно считается входящим в глобальное пространство имен. Чтобы явно указать глобальный идентификатор, используйте оператор разрешения области видимости без имени. Это позволит отличать данный идентификатор от любого другого элемента с таким же именем, находящегося в другом пространстве имен. Например, при импорте функции:

```
#import "lib.dll"
int Func();
#import
//+------------------------------------------------------------------+
//|  Некая наша функция                                              |
//+------------------------------------------------------------------+
int Func()
  {
   return(0);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//+--- вызов импортированной функции
   Print(lib::Func());
//+--- вызов нашей функции
   Print(::Func());
  }
```

В данном случае все импортированные из DLL функции были включены в одноименное пространство имен. Это позволило компилятору однозначно определить какую функцию требуется вызвать.

Смотри также

© 2000-2025, MetaQuotes Ltd.

Глобальные переменные, Локальные переменные, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

Стандартные константы, перечисления и структуры

Для облегчения написания программ, а также для удобства восприятия исходных текстов программ, в языке MQL5 предусмотрены предопределенные стандартные константы и перечисления. Кроме того, для хранения информации используются служебные структуры.

Стандартные константы являются аналогом макроподстановок и имеют тип int.

Константы сгруппированы по своему назначению:

- Константы графиков используются при работе с ценовыми графиками: открытие, навигация, установка параметров;
- Константы объектов предназначены для обработки графических объектов, которые можно создавать и отображать на графиках;
- Константы индикаторов служат для работы со стандартными и пользовательскими индикаторами;
- Состояние окружения – описывают свойства mql5-программы, предоставляют информацию о клиентском терминале, торговом инструменте и текущем торговом счете;
- Торговые константы позволяют уточнять разнообразную информацию в процессе торговли;
- Именованные константы – константы языка MQL5;
- Структуры данных описывают используемые форматы хранения данных;
- Коды ошибок и предупреждений описывают сообщения компилятора и сообщения торгового сервера на торговые запросы;
- Константы ввода/вывода предназначены для работы с файловыми функциями и вывода сообщений на экран компьютера функцией MessageBox().

© 2000-2025, MetaQuotes Ltd.

Константы графиков

Константы, описывающие различные свойства графиков, разделены на следующие группы:

- Типы событий – события, которые возникают при работе с графиком;
- Периоды графиков – стандартные встроенные периоды;
- Свойства графиков – идентификаторы, используемые в качестве параметров функций для работы с графиками;
- Константы позиционирования - значения параметра функции ChartNavigate();
- Отображение графиков - задание внешнего вида графика.

© 2000-2025, MetaQuotes Ltd.

Типы событий графика

Существуют 11 видов событий, которые можно обрабатывать с помощью функции предопределенной функции OnChartEvent(). Для пользовательских событий предусмотрено 65535 идентификаторов в диапазоне от CHARTEVENT_CUSTOM до CHARTEVENT_CUSTOM_LAST включительно. Для генерации пользовательского события необходимо использовать функцию EventChartCustom().

ENUM_CHART_EVENT

Идентификатор	Описание
CHARTEVENT_KEYDOWN	Нажатие клавиатуры
CHARTEVENT_MOUSE_MOVE	Перемещение мыши и нажатие кнопок мышки (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE=true)
CHARTEVENT_MOUSE_WHEEL	Нажатие или прокрутка колесика мышки (если для графика установлено свойство CHART_EVENT_MOUSE_WHEEL=true)
CHARTEVENT_OBJECT_CREATE	Создание графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_CREATE=true)
CHARTEVENT_OBJECT_CHANGE	Изменение свойств графического объекта через диалог свойств
CHARTEVENT_OBJECT_DELETE	Удаление графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_DELETE=true)
CHARTEVENT_CLICK	Нажатие мышки на графике
CHARTEVENT_OBJECT_CLICK	Нажатие мышки на графическом объекте
CHARTEVENT_OBJECT_DRAG	Перетаскивание графического объекта
CHARTEVENT_OBJECT_ENDEDIT	Окончание редактирования текста в графическом объекте Edit
CHARTEVENT_CHART_CHANGE	Изменение размеров графика или изменение свойств графика через диалог свойств
CHARTEVENT_CUSTOM	Начальный номер события из диапазона пользовательских событий
CHARTEVENT_CUSTOM_LAST	Конечный номер события из диапазона пользовательских событий

Для каждого типа события входные параметры функции OnChartEvent() имеют определенные значения, которые необходимы для обработки этого события. В таблице перечислены события и значения, которые передаются через параметры.

© 2000-2025, MetaQuotes Ltd.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
Событие нажатия клавиатуры	CHARTEVENT_KEYDOWN	код нажатой клавиши	Количество нажатий клавиши,сгенерированных за время её удержания в нажатом состоянии	Строковое значение битовой маски, описывающее статус кнопок клавиатуры
События мышки (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	X координата	Y координата	Строковое значение битовой маски, описывающее статус кнопок мыши
Событие колёсика мышки (если для графика установлено свойство CHART_EVENT_MOUSE_WHEEL=true)	CHARTEVENT_MOUSE_WHEEL	Флаги состояний клавиш и кнопок мышки, координаты X и Y курсора. Описание дано в примере ниже	Значение Delta прокрутки колёсика мышки	—
Событие создания графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Имя созданного графического объекта
Событие изменения свойств объекта через диалог свойств	CHARTEVENT_OBJECT_CHANGE	—	—	Имя измененного графического объекта
Событие удаления графического объекта (если	CHARTEVENT_OBJECT_DELETE	—	—	Имя удаленного графического объекта

© 2000-2025, MetaQuotes Ltd.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
для графика установлено свойство CHART_EVENT_OBJECT_DELETE=true)				
Событие щелчка мышки на графике	CHARTEVENT_CLICK	X координата	Y координата	—
Событие щелчка мышки на графическом объекте	CHARTEVENT_OBJECT_CLICK	X координата	Y координата	Имя графического объекта, на котором произошло событие
Событие перемещения графического объекта при помощи мышки	CHARTEVENT_OBJECT_DRAG	—	—	Имя перемещенного графического объекта
Событие окончания редактирования текста в поле ввода графического объекта "Поле ввода"	CHARTEVENT_OBJECT_ENDEDIT	—	—	Имя графического объекта "Поле ввода", в котором завершилось редактирование текста
Событие изменения размеров графика или изменения свойств графика через диалог свойств	CHARTEVENT_CHART_CHANGE	—	—	—
Пользовательское событие с номером N	CHARTEVENT_CUSTOM+N	Значение, заданное функцией EventChartCustom()	Значение, заданное функцией EventChartCustom()	Значение, заданное функцией EventChartCustom()

Пример:

© 2000-2025, MetaQuotes Ltd.

```
#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()
  {
//---
   Print("Запущен эксперт с именем ",MQL5InfoString(MQL5_PROGRAM_NAME));
//--- установка флага получения событий создания объектов графика
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- установка флага получения событий удаления объектов графика
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- принудительное обновление свойств графика гарантирует готовность к обработке соб
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // идентификатор события
                  const long& lparam,   // параметр события типа long
                  const double& dparam, // параметр события типа double
                  const string& sparam  // параметр события типа string
                  )
  {
//--- нажатие левой кнопкой мышки на графике
   if(id==CHARTEVENT_CLICK)
      Print("Координаты щелчка мышки на графике: x = ",lparam,"  y = ",dparam);
//--- нажатие мышкой на графическом объекте
   if(id==CHARTEVENT_OBJECT_CLICK)
      Print("Нажатие кнопки мышки на объекте с именем '"+sparam+"'");
//--- нажатие кнопки на клавиатуре
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch(lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("Нажата KEY_NUMLOCK_LEFT");   break;
         case KEY_LEFT:          Print("Нажата KEY_LEFT");           break;
         case KEY_NUMLOCK_UP:    Print("Нажата KEY_NUMLOCK_UP");     break;
```

© 2000-2025, MetaQuotes Ltd.

```
         case KEY_UP:             Print("Нажата KEY_UP");             break;
         case KEY_NUMLOCK_RIGHT: Print("Нажата KEY_NUMLOCK_RIGHT"); break;
         case KEY_RIGHT:          Print("Нажата KEY_RIGHT");          break;
         case KEY_NUMLOCK_DOWN:  Print("Нажата KEY_NUMLOCK_DOWN");   break;
         case KEY_DOWN:           Print("Нажата KEY_DOWN");           break;
         case KEY_NUMPAD_5:       Print("Нажата KEY_NUMPAD_5");       break;
         case KEY_NUMLOCK_5:      Print("Нажата KEY_NUMLOCK_5");      break;
         default:                 Print("Нажата какая-то неперечисленная клавиша");
        }
      ChartRedraw();
     }
//--- удален объекта
   if(id==CHARTEVENT_OBJECT_DELETE)
      Print("Удален объект с именем ",sparam);
//--- создан объект
   if(id==CHARTEVENT_OBJECT_CREATE)
      Print("Создан объект с именем ",sparam);
//--- перемещен объект или изменены координаты точек привязки
   if(id==CHARTEVENT_OBJECT_DRAG)
      Print("Изменение точек привязки объекта с именем ",sparam);
//--- изменен текст в поле ввода графического объекта Edit
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      Print("Изменен текст в объекте Edit ",sparam);
  }
```

Для события CHARTEVENT_MOUSE_MOVE строковой параметр **sparam** содержит число, представляющее информацию о состоянии клавиш:

Бит	Описание
1	Состояние левой клавиши мыши
2	Состояние правой клавиши мыши
3	Состояние клавиши SHIFT
4	Состояние клавиши CTRL
5	Состояние средней клавиши мыши
6	Состояние первой дополнительной клавиши мыши
7	Состояние второй дополнительной клавиши мыши

Пример:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- включение сообщений о перемещении мыши по окну чарта
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- отключаем контекстное меню чарта (по правой кнопке)
   ChartSetInteger(0,CHART_CONTEXT_MENU,0);
//--- отключаем перекрестие (по средней кнопке)
   ChartSetInteger(0,CHART_CROSSHAIR_TOOL,0);
//--- принудительное обновление свойств графика гарантирует готовность к обработке соб
   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));
  }
```

```
//--- отключаем контекстное меню чарта (по правой кнопке)
   ChartSetInteger(0,CHART_CONTEXT_MENU,0);
//--- отключаем перекрестие (по средней снопке)
   ChartSetInteger(0,CHART_CROSSHAIR_TOOL,0);
```

Для события CHARTEVENT_MOUSE_WHEEL параметры **lparam** и **dparam** содержат информацию о состоянии клавиш **Ctrl**, **Shift**, кнопок мышки, координатах курсора и величине прокрутки колёсика мышки. Для лучшего понимания запустите этот советник на графике и прокручивайте колёсико мышки, нажимая поочередно различные кнопки и клавиши, описанные в коде.

Пример обработки события CHARTEVENT_MOUSE_WHEEL:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
init OnInit()
  {
//--- включение сообщений о прокрутке колесика мышки
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- принудительное обновление свойств графика гарантирует готовность к обработке соб
   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)
     {
      //--- разберем состояние кнопок и колесика мышки для этого события
      int flg_keys = (int)(lparam>>32);          // флаг состояний клавиш Ctrl, Shift
      int x_cursor = (int)(short)lparam;         // X-координата, в которой произошло
      int y_cursor = (int)(short)(lparam>>16);   // Y-координата, в которой произошло
      int delta    = (int)dparam;                // суммарное значение прокрутки колес
      //--- обработаем флаг
      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_
     }
  }
//+------------------------------------------------------------------+ /*
```

Пример вывода

```
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
```

© 2000-2025, MetaQuotes Ltd.

```
   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  */
```

Смотри также

Функции обработки событий, Работа с событиями

© 2000-2025, MetaQuotes Ltd.

Периоды графиков

Все предопределенные периоды графиков имеют уникальные идентификаторы. Идентификатор PERIOD_CURRENT означает текущий период графика, на котором запущена mql5-программа.

ENUM_TIMEFRAMES

Идентификатор	Описание
PERIOD_CURRENT	Текущий период
PERIOD_M1	1 минута
PERIOD_M2	2 минуты
PERIOD_M3	3 минуты
PERIOD_M4	4 минуты
PERIOD_M5	5 минут
PERIOD_M6	6 минут
PERIOD_M10	10 минут
PERIOD_M12	12 минут
PERIOD_M15	15 минут
PERIOD_M20	20 минут
PERIOD_M30	30 минут
PERIOD_H1	1 час
PERIOD_H2	2 часа
PERIOD_H3	3 часа
PERIOD_H4	4 часа
PERIOD_H6	6 часов
PERIOD_H8	8 часов
PERIOD_H12	12 часов
PERIOD_D1	1 день
PERIOD_W1	1 неделя
PERIOD_MN1	1 месяц

Пример:

```
   string chart_name="test_Object_Chart";
   Print("Попробуем создать объект Chart  с именем ",chart_name);
//--- если такого объекта нет - создадим его
   if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- зададим символ
   ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- зададим координату X для точки привязки
   ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- зададим координату Y для точки привязки
   ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- установим ширину
   ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- установим высоту
   ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- установим период
   ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- установим масшаб ( от 0 до 5)
   ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- сделаем недоступным для выделения мышкой
   ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

Идентификаторы таймсерий

Идентификаторы таймсерий используются в функциях iHighest() и iLowest(). Могут быть одним из значений перечисления

ENUM_SERIESMODE

Идентификатор	Описание
MODE_OPEN	Цена открытия
MODE_LOW	Минимальная цена
MODE_HIGH	Максимальная цена
MODE_CLOSE	Цена закрытия
MODE_VOLUME	Тиковый объем
MODE_REAL_VOLUME	Реальный объем
MODE_SPREAD	Спред

Смотри также

PeriodSeconds, Period, Дата и время, Видимость объектов

© 2000-2025, MetaQuotes Ltd.

Свойства графиков

Идентификаторы семейства перечислений ENUM_CHART_PROPERTY используются в качестве параметров функций для работы с графиками. Аббревиатура r/o в столбце "Тип свойства" означает, что данное свойство предназначено только для чтения и не может быть изменено. Аббревиатура w/o в столбце "Тип свойства" означает, что данное свойство предназначено только для записи и не может быть получено. При обращении к некоторым свойствам необходимо указывать дополнительный параметр-модификатор (modifier), который служит для указания номера подокна графика. 0 означает главное окно.

Функции, устанавливающие свойства графика, фактически служат для отправки ему команд на изменение. При успешном выполнении этих функций команда попадает в общую очередь событий графика. Изменение графика производится в процессе обработки очереди событий данного графика.

По этой причине не следует ожидать немедленного визуального обновления графика после вызова данных функций. В общем случае обновление графика производится терминалом автоматически по событиям изменения - поступление новой котировки, изменения размера окна графика и т.д. Для принудительного обновления внешнего вида графика используйте команду на перерисовку графика ChartRedraw().

Для функций ChartSetInteger() и ChartGetInteger()

ENUM_CHART_PROPERTY_INTEGER

Идентификатор	Описание	Тип свойства
CHART_SHOW	Признак отрисовки ценового графика. Если установлено значение false, то отключается отрисовка любых атрибутов ценового графика и устраняются все отступы по краям графика: шкалы времени и цены, строка быстрой навигации, метки событий Календаря, значки сделок, тултипы индикаторов и	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	баров, подокна индикаторов, гистограммы объёмов и т.д. Отключение отрисовки является идеальным решением для создания собственного интерфейса программы с использованием графических ресурсов. Графические объекты отрисовываются всегда независимо от установленного значения свойства CHART_SHOW.	
CHART_IS_OBJECT	Признак идентификации объекта "График" (OBJ_CHART) - для графического объекта возвращает true. Для настоящего графика значение равно false	bool r/o
CHART_BRING_TO_TOP	Показ графика поверх всех других	bool
CHART_CONTEXT_MENU	Включение/отключение доступа к контекстному меню по нажатию правой	bool (значение по умолчанию true)

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	клавиши мышки. Значение CHART_CONTEXT_MENU=false отключает только контекстное меню графика, при этом контекстное меню для объектов на графике остается доступным.	
CHART_CROSSHAIR_TOOL	Включение/отключение доступа к инструменту "Перекрестие" по нажатию средней клавиши мышки	bool (значение по умолчанию true)
CHART_MOUSE_SCROLL	Прокрутка графика левой кнопкой мышки по горизонтали. Прокрутка по вертикали также будет доступна, если установлено в true значение любого из трех свойств: CHART_SCALEFIX, CHART_SCALEFIX_11 или CHART_SCALE_PT_PER_BAR При CHART_MOUSE_SCROLL=false прокрутка графика колёсиком	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	мышки будет недоступна	
CHART_EVENT_MOUSE_WHEEL	Отправка всем mql5-программам на графике сообщений о событиях колёсика мышки (CHARTEVENT_MOUSE_WHEEL)	bool (значение по умолчанию true)
CHART_EVENT_MOUSE_MOVE	Отправка всем mql5-программам на графике сообщений о событиях перемещения и нажатия кнопок мышки (CHARTEVENT_MOUSE_MOVE)	bool
CHART_EVENT_OBJECT_CREATE	Отправка всем mql5-программам на графике сообщений о событии создания графического объекта (CHARTEVENT_OBJECT_CREATE)	bool
CHART_EVENT_OBJECT_DELETE	Отправка всем mql5-программам на графике сообщений о событии уничтожения графического объекта (CHARTEVENT_OBJECT_DELETE)	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
CHART_MODE	Тип графика (свечи, бары или линия)	enum ENUM_CHART_MODE
CHART_FOREGROUND	Ценовой график на переднем плане	bool
CHART_SHIFT	Режим отступа ценового графика от правого края	bool
CHART_AUTOSCROLL	Режим автоматического перехода к правому краю графика	bool
CHART_KEYBOARD_CONTROL	Разрешение на управление графиком с помощью клавиатуры ("Home", "End", "PageUp", "+", "-", "Стрелка вверх" и т.д.). Установка CHART_KEYBOARD_CONTROL=false позволяет отключить скроллинг и масштабирование графика, но при этом сохраняется возможность получать события нажатия данных клавиш в OnChartEvent().	bool
CHART_QUICK_NAVIGATION	Разрешение на перехват графиком нажатий клавиш Space и Enter для активации	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	строки быстрой навигации. Строка быстрой навигации автоматически появляется внизу графика при двойном клике мышки или нажатии клавиш Space/Enter. Таким образом можно быстро сменить символ, таймфрейм и дату первого видимого бара.	
CHART_SCALE	Масштаб	int от 0 до 5
CHART_SCALEFIX	Режим фиксированного масштаба	bool
CHART_SCALEFIX_11	Режим масштаба 1:1	bool
CHART_SCALE_PT_PER_BAR	Режим указания масштаба в пунктах на бар	bool
CHART_SHOW_TICKER	Отображение в левом верхнем углу тикера символа. Установка CHART_SHOW_TICKER в значение false одновременно также устанавливает CHART_SHOW_OHLC в значение false и отключает показ OHLC	bool
CHART_SHOW_OHLC	Отображение в левом верхнем	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	углу значений OHLC. Установка CHART_SHOW_OHLC в значение true одновременно также устанавливает CHART_SHOW_TICKER в значение true и включает показ тикера	
CHART_SHOW_BID_LINE	Отображение значения Bid горизонтальной линией на графике	bool
CHART_SHOW_ASK_LINE	Отображение значения Ask горизонтальной линией на графике	bool
CHART_SHOW_LAST_LINE	Отображение значения Last горизонтальной линией на графике	bool
CHART_SHOW_PERIOD_SEP	Отображение вертикальных разделителей между соседними периодами	bool
CHART_SHOW_GRID	Отображение сетки на графике	bool
CHART_SHOW_VOLUMES	Отображение объемов на графике	enum ENUM_CHART_VOLUME_MODE
CHART_SHOW_OBJECT_DESCR	Отображение текстовых описаний	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	объектов (не для всех объектов описания показываются)	
CHART_SHOW_TRADE_HISTORY	Отображение на графике сделок из торговой истории в виде стрелок входов/выходов. Смотрите описание опции "Показать торговую историю" в настройках платформы.	bool
CHART_VISIBLE_BARS	Количество баров на графике, доступных для отображения	int r/o
CHART_WINDOWS_TOTAL	Общее количество окон графика, включая подокна индикаторов	int r/o
CHART_WINDOW_IS_VISIBLE	Видимость подокон	bool r/o модификатор - номер подокна
CHART_WINDOW_HANDLE	Хэндл графика (HWND)	int r/o
CHART_WINDOW_YDISTANCE	Дистанция в пикселях по вертикальной оси Y между верхней рамкой подокна индикатора и верхней рамкой главного окна графика. При наступлении событий мыши	int r/o модификатор - номер подокна

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	координаты курсора передаются в координатах главного окна графика, в то время как координаты графических объектов в подокне индикатора задаются относительно верхнего левого угла подокна. Значение требуется для перевода абсолютных координат главного графика в локальные координаты подокна для корректной работы с графическими объектами, у которых координаты задаются относительно верхнего левого угла рамки подокна.	
CHART_FIRST_VISIBLE_BAR	Номер первого видимого бара на графике. Индексация баров соответствует таймсерии.	int r/o
CHART_WIDTH_IN_BARS	Ширина графика в барах	int r/o

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
CHART_WIDTH_IN_PIXELS	Ширина графика в пикселях	int r/o
CHART_HEIGHT_IN_PIXELS	Высота графика в пикселях	int модификатор - номер подокна
CHART_COLOR_BACKGROUND	Цвет фона графика	color
CHART_COLOR_FOREGROUND	Цвет осей, шкалы и строки OHLC	color
CHART_COLOR_GRID	Цвет сетки	color
CHART_COLOR_VOLUME	Цвет объемов и уровней открытия позиций	color
CHART_COLOR_CHART_UP	Цвет бара вверх, тени и окантовки тела бычьей свечи	color
CHART_COLOR_CHART_DOWN	Цвет бара вниз, тени и окантовки тела медвежьей свечи	color
CHART_COLOR_CHART_LINE	Цвет линии графика и японских свечей "Доджи"	color
CHART_COLOR_CANDLE_BULL	Цвет тела бычьей свечи	color
CHART_COLOR_CANDLE_BEAR	Цвет тела медвежьей свечи	color
CHART_COLOR_BID	Цвет линии Bid-цены	color
CHART_COLOR_ASK	Цвет линии Ask-цены	color
CHART_COLOR_LAST	Цвет линии цены последней совершенной сделки (Last)	color

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
CHART_COLOR_STOP_LEVEL	Цвет уровней стоп-ордеров (Stop Loss и Take Profit)	color
CHART_SHOW_TRADE_LEVELS	Отображение на графике торговых уровней (уровни открытых позиций, Stop Loss, Take Profit и отложенных ордеров)	bool
CHART_DRAG_TRADE_LEVELS	Разрешение на перетаскивание торговых уровней на графике с помощью мышки. По умолчанию режим перетаскивания включен (значение true)	bool
CHART_SHOW_DATE_SCALE	Отображение на графике шкалы времени	bool
CHART_SHOW_PRICE_SCALE	Отображение на графике ценовой шкалы	bool
CHART_SHOW_ONE_CLICK	Отображение на графике панели быстрой торговли (опция "Торговля одним кликом")	bool
CHART_IS_MAXIMIZED	Окно графика развернуто	bool r/o
CHART_IS_MINIMIZED	Окно графика свернуто	bool r/o
CHART_IS_DOCKED	Окно графика закреплено. Если установить	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	значение false, то график можно перетащить за пределы терминала	
CHART_FLOAT_LEFT	Левая координата откреплённого графика относительно виртуального экрана	int
CHART_FLOAT_TOP	Верхняя координата откреплённого графика относительно виртуального экрана	int
CHART_FLOAT_RIGHT	Правая координата откреплённого графика относительно виртуального экрана	int
CHART_FLOAT_BOTTOM	Нижняя координата откреплённого графика относительно виртуального экрана	int

Для функций ChartSetDouble() и ChartGetDouble()

ENUM_CHART_PROPERTY_DOUBLE

Идентификатор	Описание	Тип свойства
CHART_SHIFT_SIZE	Размер отступа нулевого бара от правого края в процентах	double (от 10 до 50 процентов)

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
CHART_FIXED_POSITION	Положение фиксированной позиции графика от левого края в процентах. Фиксированная позиция графика обозначена маленьким серым треугольником на горизонтальной оси времени и показывается только в том случае, если отключена автоматическая прокрутка к правому краю при поступлении нового тика (смотри свойство CHART_AUTOSCROLL). Бар, который находится на фиксированной позиции, остаётся на том же месте при увеличении и уменьшении масштаба.	double
CHART_FIXED_MAX	Фиксированный максимум графика	double
CHART_FIXED_MIN	Фиксированный минимум графика	double
CHART_POINTS_PER_BAR	Значение масштаба в пунктах на бар	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
CHART_PRICE_MIN	Минимум графика	double r/o модификатор - номер подокна
CHART_PRICE_MAX	Максимум графика	double r/o модификатор - номер подокна

Для функций ChartSetString() и ChartGetString()

ENUM_CHART_PROPERTY_STRING

Идентификатор	Описание	Тип свойства
CHART_COMMENT	Текст комментария на графике	string
CHART_EXPERT_NAME	Имя эксперта, запущенного на графике с указанным chart_id	string r/o
CHART_SCRIPT_NAME	Имя скрипта, запущенного на графике с указанным chart_id	string r/o

Пример:

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   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);
        }
     }
```

Смотри также

Примеры работы с графиком

© 2000-2025, MetaQuotes Ltd.

Позиционирование графика

Три идентификатора из списка ENUM_CHART_POSITION являются возможными значениями параметра position для функции ChartNavigate().

ENUM_CHART_POSITION

Идентификатор	Описание
CHART_BEGIN	Начало графика (самые старые цены)
CHART_CURRENT_POS	Текущая позиция
CHART_END	Конец графика (самые свежие цены)

Пример:

```
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("Navigate failed. Error = ",GetLastError());
   ChartRedraw();
  }
```

© 2000-2025, MetaQuotes Ltd.

Отображение графиков

Ценовые графики можно отображать тремя способами:

- в виде баров;
- в виде свечей;
- в виде ломаной линии.

Конкретный способ отображения ценового графика задается функцией ChartSetInteger(handle_графика,CHART_MODE, тип_графика), где тип_графика - одно из значений перечисления ENUM_CHART_MODE.

ENUM_CHART_MODE

Идентификатор	Описание
CHART_BARS	Отображение в виде баров
CHART_CANDLES	Отображение в виде японских свечей
CHART_LINE	Отображение в виде линии, проведенной по ценам Close

Для указания режима отображения объемов на ценовом графике используется функция ChartSetInteger(handle_графика, CHART_SHOW_VOLUMES, тип_отображения), где тип_отображения - одно из значений перечисления ENUM_CHART_VOLUME_MODE.

ENUM_CHART_VOLUME_MODE

Идентификатор	Описание
CHART_VOLUME_HIDE	Объемы не показаны
CHART_VOLUME_TICK	Тиковые объемы
CHART_VOLUME_REAL	Торговые объемы

Пример:

```
//--- получим handle текущего графика
   long handle=ChartID();
   if(handle>0) // если получилось, дополнительно настроим
     {
      //--- отключим автопрокрутку
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- установим отступ правого края графика
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- отобразим в виде свечей
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- прокрутим на 100 баров от начала истории
      ChartNavigate(handle,CHART_CURRENT_POS,100);
      //--- установить режим отображения тиковых объемов
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
     }
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

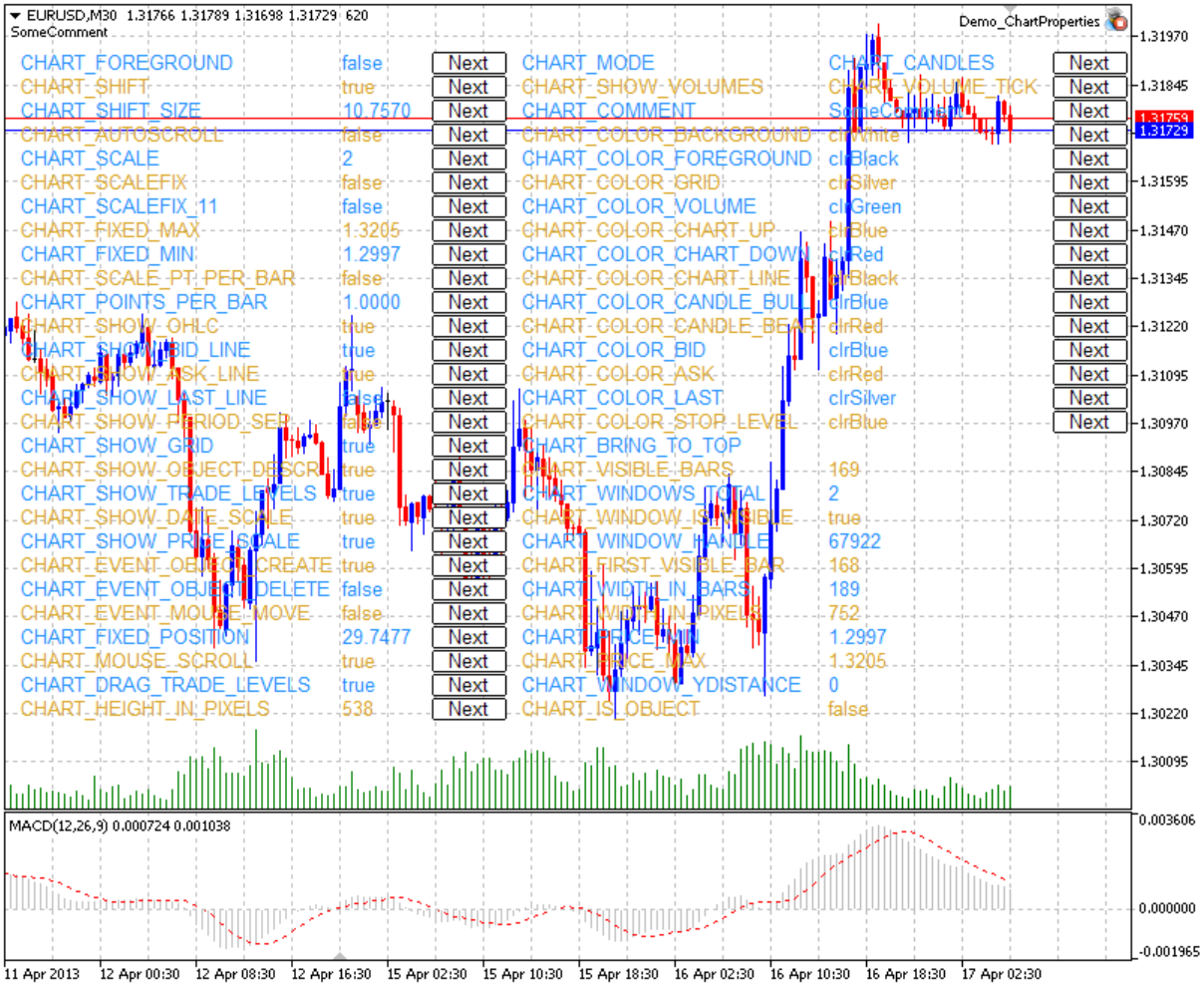
ChartOpen, ChartID

© 2000-2025, MetaQuotes Ltd.

Примеры работы с графиком

В этом разделе представлены примеры для работы со свойствами графика. Для каждого свойства приведены одна или две законченные функции, которые позволяют задавать/получать значение этого свойства. Эти функции можно использовать в своих MQL5 программах как есть.

На рисунке показана графическая панель для наглядной демонстрации того, как изменение свойства графика меняет его внешний вид. Нажатие кнопки "Next" позволяет установить новое значение соответствующего свойства и увидеть изменения в окне графика.



Исходный код панели находится ниже.

Свойства графика и примеры функций для работы с ними

- **CHART_IS_OBJECT** - определяет, является ли объект настоящим графиком или графическим объектом.

```
//+------------------------------------------------------------------+
//| Определение того, является ли объект графиком. Если это          |
//| графический объект, то результат true. Если же это настоящий     |
//| график, то переменная result примет значение false.              |
//+------------------------------------------------------------------+
bool ChartIsObject(bool &result,const long chart_ID=0)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим свойство графика
   if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      //--- вернем false
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
```

- **CHART_BRING_TO_TOP** - показывает график поверх всех других.

```
//+------------------------------------------------------------------+
//| Отправка терминалу команды на показ графика поверх всех других.  |
//+------------------------------------------------------------------+
bool ChartBringToTop(const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- покажем график поверх всех других
   if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_MOUSE_SCROLL** - свойство прокрутки графика левой кнопкой мыши.

```
//+------------------------------------------------------------------+
//| Функция определяет, можно ли прокручивать график левой кнопкой   |
//| мыши.                                                            |
//+------------------------------------------------------------------+
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим прокрутки графика левой кнопкой |
//| мыши.                                                            |
//+------------------------------------------------------------------+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_EVENT_MOUSE_MOVE** - свойство отправки mql5-программам сообщений о событиях перемещения и нажатия кнопок мыши (CHARTEVENT_MOUSE_MOVE).

```
//+------------------------------------------------------------------+
//| Отправляются ли сообщения о событиях перемещения и нажатия кнопок|
//| мыши всем mql5-программам на данном графике?                     |
//+------------------------------------------------------------------+
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отправки сообщений о событиях   |
//| перемещения и нажатия кнопок мыши всем mql5-программам на данном |
//| графике.                                                         |
//+------------------------------------------------------------------+
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_EVENT_OBJECT_CREATE** - свойство отправки mql5-программам сообщений о событии создания графического объекта (CHARTEVENT_OBJECT_CREATE).

```
//+------------------------------------------------------------------+
//| Отправляются ли сообщения о событии создания графического объекта|
//| всем mql5-программам на данном графике?                          |
//+------------------------------------------------------------------+
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отправки сообщений о событии    |
//| создания графического объекта всем mql5-программам на данном     |
//| графике.                                                         |
//+------------------------------------------------------------------+
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_EVENT_OBJECT_DELETE** - свойство отправки mql5-программам сообщений о событии удаления графического объекта (CHARTEVENT_OBJECT_DELETE).

```
//+------------------------------------------------------------------+
//| Отправляются ли сообщения о событии удаления графического объекта|
//| всем mql5-программам на данном графике?                          |
//+------------------------------------------------------------------+
bool ChartEventObjectDeleteGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отправки сообщений о событии    |
//| удаления графического объекта всем mql5-программам на данном     |
//| графике.                                                         |
//+------------------------------------------------------------------+
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_MODE** - тип графика (свечи, бары или линия).

```
//+------------------------------------------------------------------+
//| Получение типа отображения графика (в виде свечей, баров или     |
//| линии).                                                          |
//+------------------------------------------------------------------+
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=WRONG_VALUE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((ENUM_CHART_MODE)result);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Установка типа отображения графика (в виде свечей, баров или     |
//| линии).                                                           |
//+------------------------------------------------------------------+
bool ChartModeSet(const long value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_MODE,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_FOREGROUND** - свойство отображения ценового графика на переднем плане.

```
//+------------------------------------------------------------------+
//| Функция определяет, отображается ли ценовой график на переднем   |
//| плане.                                                           |
//+------------------------------------------------------------------+
bool ChartForegroundGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения ценового графика на |
//| переднем плане.                                                  |
//+------------------------------------------------------------------+
bool ChartForegroundSet(const bool value,const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHIFT** - режим отступа ценового графика от правого края.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим отображения ценового графика|
//| с отступом от правого края.                                      |
//+------------------------------------------------------------------+
bool ChartShiftGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения ценового графика с  |
//| отступом от правого края.                                        |
//+------------------------------------------------------------------+
bool ChartShiftSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_AUTOSCROLL** - режим автоматического перехода к правому краю графика.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим автоматической прокрутки    |
//| графика вправо при поступлении новых тиков.                      |
//+------------------------------------------------------------------+
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим автоматической прокрутки графика|
//| вправо при поступлении новых тиков.                              |
//+------------------------------------------------------------------+
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
```

- **CHART_SCALE** - свойство масштаба графика.

```
//+------------------------------------------------------------------+
//| Получение масштаба графика (от 0 до 5).                          |
//+------------------------------------------------------------------+
int ChartScaleGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((int)result);
  }
//+------------------------------------------------------------------+
//| Установка масштаба графика (от 0 до 5).                          |
//+------------------------------------------------------------------+
bool ChartScaleSet(const long value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SCALEFIX** - режим фиксированного масштаба графика.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим фиксированного масштаба.    |
//+------------------------------------------------------------------+
bool ChartScaleFixGet(bool &result,const long chart_ID=0)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим фиксированного масштаба.       |
//+------------------------------------------------------------------+
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SCALEFIX_11** - режим масштаба графика 1:1.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим масштаба "1:1".             |
//+------------------------------------------------------------------+
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- выведем сообщение об ошибке в журнал "Эксперты"
       Print(__FUNCTION__+", Error Code = ",GetLastError());
       return(false);
      }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим масштаба "1:1"                  |
//+------------------------------------------------------------------+
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SCALE_PT_PER_BAR** - режим указания масштаба графика в пунктах на бар.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим указания масштаба в пунктах |
//| на бар.                                                          |
//+------------------------------------------------------------------+
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
```

© 2000-2025, MetaQuotes Ltd.

```
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим указания масштаба в пунктах на |
//| бар.                                                             |
//+------------------------------------------------------------------+
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_OHLC** - свойство отображения в левом верхнем углу значений OHLC.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим отображения значений OHLC   |
//| в левом верхнем углу.                                            |
//+------------------------------------------------------------------+
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Функция включает/выключает режим отображения значений OHLC в      |
//| левом верхнем углу графика.                                       |
//+------------------------------------------------------------------+
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_BID_LINE** - свойство отображения значения Bid горизонтальной линией на графике.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим отображения линии Bid на     |
//| графике.                                                          |
//+------------------------------------------------------------------+
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения линии Bid на         |
//| графике.                                                          |
//+------------------------------------------------------------------+
bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_ASK_LINE** - свойство отображения значения Ask горизонтальной линией на графике.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим отображения линии Ask на    |
//| графике.                                                         |
//+------------------------------------------------------------------+
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения линии Ask на        |
//| графике.                                                         |
//+------------------------------------------------------------------+
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_LAST_LINE** - свойство отображения значения Last горизонтальной линией на графике.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим отображения линии для цены  |
//| последней совершенной сделки.                                    |
//+------------------------------------------------------------------+
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения линии цены последней|
//| совершенной сделки.                                              |
//+------------------------------------------------------------------+
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_PERIOD_SEP** - свойство отображения вертикальных разделителей между соседними периодами.

```
//+------------------------------------------------------------------+
//| Функция определяет, включен ли режим отображения вертикальных    |
//| разделителей между соседними периодами.                          |
//+------------------------------------------------------------------+
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения вертикальных        |
//| разделителей между соседними периодами.                          |
//+------------------------------------------------------------------+
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
```

- **CHART_SHOW_GRID** - свойство отображения сетки на графике.

```
//+------------------------------------------------------------------+
//| Функция определяет, отображается ли сетка на графике.            |
//+------------------------------------------------------------------+
bool ChartShowGridGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает отображение сетки на графике.         |
//+------------------------------------------------------------------+
bool ChartShowGridSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_VOLUMES** - свойство отображения объемов на графике.

```
//+------------------------------------------------------------------+
//| Функция определяет, отображаются ли объемы на графике (не        |
```

© 2000-2025, MetaQuotes Ltd.

```
//| отображаются, отображаются тиковые, отображаются реальные).      |
//+------------------------------------------------------------------+
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=WRONG_VALUE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((ENUM_CHART_VOLUME_MODE)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает режим отображения объемов на графике.      |
//+------------------------------------------------------------------+
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_OBJECT_DESCR** - свойство всплывающих описаний графических объектов.

```
//+------------------------------------------------------------------+
//| Функция определяет, отображаются ли всплывающие описания         |
//| графических объектов при наведение на них мышью.                 |
//+------------------------------------------------------------------+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения всплывающих описаний|
//| графических объектов при наведение на них мышью.                 |
//+------------------------------------------------------------------+
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_VISIBLE_BARS** - определяет количество баров на графике, доступных для отображения.

```
//+------------------------------------------------------------------+
//| Функция получает количество баров, которые отображаются (видимы )|
//| в окне графика.                                                  |
//+------------------------------------------------------------------+
int ChartVisibleBars(const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- вернем значение свойства графика
   return((int)result);
  }
```

- **CHART_WINDOWS_TOTAL** - определяет общее количество окон графика, включая подокна индикаторов.

```
//+------------------------------------------------------------------+
//| Функция получает общее количество окон графика, включая подокна  |
//| индикаторов.                                                     |
//+------------------------------------------------------------------+
int ChartWindowsTotal(const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((int)result);
  }
```

- **CHART_WINDOW_IS_VISIBLE** - определяет видимость подокна.

```
//+------------------------------------------------------------------+
//| Функция определяет, является ли данное окно или подокно графика  |
//| видимым.                                                         |
//+------------------------------------------------------------------+
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int sub_window=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
```

© 2000-2025, MetaQuotes Ltd.

```
   result=value;
//--- успешное выполнение
   return(true);
  }
```

- **CHART_WINDOW_HANDLE** - возвращает хэндл графика.

```
//+------------------------------------------------------------------+
//| Функция получает хэндл графика                                   |
//+------------------------------------------------------------------+
int ChartWindowsHandle(const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((int)result);
  }
```

- **CHART_WINDOW_YDISTANCE** - определяет дистанцию в пикселях между верхней рамкой подокна индикатора и верхней рамкой главного окна графика.

```
//+------------------------------------------------------------------+
//| Функция получает расстояние в пикселях между верхней рамкой      |
//| подокна и верхней рамкой главного окна графика.                  |
//+------------------------------------------------------------------+
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((int)result);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
```

- **CHART_FIRST_VISIBLE_BAR** - возвращает номер первого видимого бара на графике (индексация баров соответствует таймсерии).

```
//+------------------------------------------------------------------+
//| Функция получает номер первого видимого бара на графике.         |
//| Индексация как в таймсерии, последние бары имеют меньшие индексы.|
//+------------------------------------------------------------------+
int ChartFirstVisibleBar(const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((int)result);
  }
```

- **CHART_WIDTH_IN_BARS** - возвращает ширину графика в барах.

```
//+------------------------------------------------------------------+
//| Функция получает значение ширины графика в барах.                |
//+------------------------------------------------------------------+
int ChartWidthInBars(const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((int)result);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_WIDTH_IN_PIXELS** - возвращает ширину графика в пикселях.

```
//+------------------------------------------------------------------+
//| Функция получает значение ширины графика в пикселях.             |
//+------------------------------------------------------------------+
int ChartWidthInPixels(const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((int)result);
  }
```

- **CHART_HEIGHT_IN_PIXELS** - свойство высоты графика в пикселях.

```
//+------------------------------------------------------------------+
//| Функция получает значение высоты графика в пикселях.             |
//+------------------------------------------------------------------+
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
  {
//--- подготовим переменную для получения значения свойства
   long result=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((int)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает значение высоты графика в пикселях.        |
//+------------------------------------------------------------------+
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int sub_windo
  {
//--- сбросим значение ошибки
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_BACKGROUND** - цвет фона графика.

```
//+------------------------------------------------------------------+
//| Функция получает цвет фона графика.                              |
//+------------------------------------------------------------------+
color ChartBackColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет фона графика
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BACKGROUND,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет фона графика.                         |
//+------------------------------------------------------------------+
bool ChartBackColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет фона графика
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BACKGROUND,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_COLOR_FOREGROUND** - цвет осей, шкалы и строки OHLC.

```
//+------------------------------------------------------------------+
//| Функция получает цвет осей, шкалы и строки OHLC графика.         |
//+------------------------------------------------------------------+
color ChartForeColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет осей, шкалы и строки OHLC графика
   if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет осей, шкалы и строки OHLC графика.    |
//+------------------------------------------------------------------+
bool ChartForeColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет осей, шкалы и строки OHLC графика
   if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_GRID** - цвет сетки графика.

```
//+------------------------------------------------------------------+
//| Функция получает цвет сетки графика.                             |
//+------------------------------------------------------------------+
color ChartGridColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
```

© 2000-2025, MetaQuotes Ltd.

```
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет сетки графика
   if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет сетки графика.                        |
//+------------------------------------------------------------------+
bool ChartGridColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет сетки графика
   if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_VOLUME** - цвет объемов и уровней открытия позиций.

```
//+------------------------------------------------------------------+
//| Функция получает цвет отображения объемов и уровней открытия     |
//| позиций.                                                         |
//+------------------------------------------------------------------+
color ChartVolumeColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет объемов и уровней открытия позиций
   if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет отображения объемов и уровней открытия|
//| позиций.                                                         |
//+------------------------------------------------------------------+
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет объемов и уровней открытия позиций
   if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_CHART_UP** - цвет бара вверх, тени и окантовки тела бычьей свечи.

```
//+------------------------------------------------------------------+
//| Функция получает цвет бара направленного вверх, цвет тени и      |
//| окантовки тела бычьей свечи.                                     |
//+------------------------------------------------------------------+
color ChartUpColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет бара вверх, тени и окантовки тела бычьей свечи
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет бара направленного вверх, цвет тени и |
//| окантовки тела бычьей свечи.                                     |
//+------------------------------------------------------------------+
bool ChartUpColorSet(const color clr,const long chart_ID=0)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет бара вверх, тени и окантовки тела бычьей свечи
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_CHART_DOWN** - цвет бара вниз, тени и окантовки тела медвежьей свечи.

```
//+------------------------------------------------------------------+
//| Функция получает цвет бара направленного вниз, цвет тени и       |
//| окантовки тела медвежьей свечи.                                  |
//+------------------------------------------------------------------+
color ChartDownColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет бара вниз, тени и окантовки тела медвежьей свечи
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет бара направленного вниз, цвет тени и  |
//| окантовки тела медвежьей свечи.                                  |
//+------------------------------------------------------------------+
bool ChartDownColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет бара вниз, тени и окантовки тела медвежьей свечи
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_CHART_LINE** - цвет линии графика и японских свечей "Доджи".

```
//+------------------------------------------------------------------+
//| Функция получает цвет линии графика и японских свечей "Доджи".   |
//+------------------------------------------------------------------+
color ChartLineColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет линии графика и японских свечей "Доджи"
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет линии графика и японских свечей       |
//| "Доджи".                                                         |
//+------------------------------------------------------------------+
bool ChartLineColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет линии графика и японских свечей "Доджи"
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BULL** - цвет тела бычьей свечи.

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Функция получает цвет тела бычьей свечи.                         |
//+------------------------------------------------------------------+
color ChartBullColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет тела бычьей свечи
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет тела бычьей свечи.                    |
//+------------------------------------------------------------------+
bool ChartBullColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет тела бычьей свечи
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BEAR** - цвет тела медвежьей свечи.

```
//+------------------------------------------------------------------+
//| Функция получает цвет тела медвежьей свечи.                      |
//+------------------------------------------------------------------+
color ChartBearColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет тела медвежьей свечи
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет тела медвежьей свечи.                 |
//+------------------------------------------------------------------+
bool ChartBearColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет тела медвежьей свечи
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_BID** - цвет линии цены Bid.

```
//+------------------------------------------------------------------+
//| Функция получает цвет отображения линии Bid.                     |
//+------------------------------------------------------------------+
color ChartBidColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет линии цены Bid
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет отображения линии Bid.                |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ChartBidColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет линии цены Bid
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_ASK** - цвет линии цены Ask.

```
//+------------------------------------------------------------------+
//| Функция получает цвет отображения линии Ask.                     |
//+------------------------------------------------------------------+
color ChartAskColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет линии цены Ask
   if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет отображения линии Ask.                |
//+------------------------------------------------------------------+
bool ChartAskColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет линии цены Ask
   if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_LAST** - цвет линии цены последней совершенной сделки (Last).

```
//+------------------------------------------------------------------+
//| Функция получает цвет линии цены последней совершенной сделки.   |
//+------------------------------------------------------------------+
color ChartLastColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет линии цены последней совершенной сделки (Last)
   if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвет линии цены последней совершенной      |
//| сделки.                                                          |
//+------------------------------------------------------------------+
bool ChartLastColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет линии цены последней совершенной сделки (Last)
   if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_COLOR_STOP_LEVEL** - цвет уровней стоп-ордеров (Stop Loss и Take Profit).

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Функция получает цвета уровней Stop Loss и Take Profit.          |
//+------------------------------------------------------------------+
color ChartStopLevelColorGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения цвета
   long result=clrNONE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим цвет уровней стоп-ордеров (Stop Loss и Take Profit)
   if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает цвета уровней Stop Loss и Take Profit.     |
//+------------------------------------------------------------------+
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим цвет уровней стоп-ордеров (Stop Loss и Take Profit)
   if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_TRADE_LEVELS** - свойство отображения на графике торговых уровней (уровни открытых позиций, Stop Loss, Take Profit и отложенных ордеров).

```
//+------------------------------------------------------------------+
//| Функция определяет, отображаются ли на графике торговые уровни.  |
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения торговых уровней.   |
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_DRAG_TRADE_LEVELS** - свойство разрешения на перетаскивание торговых уровней на графике с помощью мышки.

```
//+------------------------------------------------------------------+
//| Функция определяет, можно ли перетаскивать торговые уровни на    |
//| графике при помощи мыши.                                         |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим перетаскивание торговых уровней |
//| на графике при помощи мыши.                                      |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_DATE_SCALE** - свойство отображения на графике шкалы времени.

```
//+------------------------------------------------------------------+
//| Функция определяет, отображается ли на графике шкала времени.    |
//+------------------------------------------------------------------+
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения шкалы времени на    |
//| графике.                                                         |
//+------------------------------------------------------------------+
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_PRICE_SCALE** - свойство отображения на графике шкалы цены.

```
//+------------------------------------------------------------------+
//| Функция определяет, отображается ли на графике шкала цены.       |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения шкалы цены на       |
//| графике.                                                         |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHOW_ONE_CLICK** - отображение на графике панели быстрой торговли (опция "Торговля одним кликом").

```
//+------------------------------------------------------------------+
//| Функция определяет, отображается ли на графике                   |
//| панель быстрой торговли ("Торговля одним кликом")                |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения                     |
//| панели быстрой торговли на графике                               |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHIFT_SIZE** - размер отступа нулевого бара от правого края в процентах.

```
//+------------------------------------------------------------------+
//| Функция получает размер отступа нулевого бара от правого края    |
//| графика в процентах (от 10% до 50%).                             |
//+------------------------------------------------------------------+
double ChartShiftSizeGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения результата
   double result=EMPTY_VALUE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return(result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает размер отступа нулевого бара от правого    |
//| края графика в процентах (от 10% до 50%). Для включения режима   |
//| отступа, нужно установить значение свойства CHART_SHIFT равным   |
//| true.                                                            |
//+------------------------------------------------------------------+
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- успешное выполнение
   return(true);
  }
```

- **CHART_FIXED_POSITION** - положение фиксированной позиции графика от левого края в процентах.

```
//+------------------------------------------------------------------+
//| Функция получает положение фиксированной позиции графика от      |
//| левого края в процентах.                                         |
//+------------------------------------------------------------------+
double ChartFixedPositionGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения результата
   double result=EMPTY_VALUE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetDouble(chart_ID,CHART_FIXED_POSITION,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return(result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает положение фиксированной позиции графика от |
//| левого края в процентах. Для того  чтобы увидеть положение       |
//| фиксированной позиции на графике, нужно предварительно задать    |
//| значение свойства CHART_AUTOSCROLL равным false.                 |
//+------------------------------------------------------------------+
bool ChartFixedPositionSet(const double value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetDouble(chart_ID,CHART_FIXED_POSITION,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_FIXED_MAX** - свойство фиксированного максимума графика.

```
//+------------------------------------------------------------------+
//| Функция получает значение фиксированного максимума графика.      |
//+------------------------------------------------------------------+
double ChartFixedMaxGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения результата
   double result=EMPTY_VALUE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return(result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает значение фиксированного максимума графика. |
//| Для того чтобы удалось изменить значение этого свойства, нужно   |
//| предварительно задать значение свойства CHART_SCALEFIX равным    |
//| true.                                                            |
//+------------------------------------------------------------------+
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_FIXED_MIN** - свойство фиксированного минимума графика.

```
//+------------------------------------------------------------------+
//| Функция получает значение фиксированного минимума графика.       |
//+------------------------------------------------------------------+
double ChartFixedMinGet(const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- подготовим переменную для получения результата
   double result=EMPTY_VALUE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return(result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает значение фиксированного минимума графика.  |
//| Для того чтобы удалось изменить значение этого свойства, нужно   |
//| предварительно задать значение свойства CHART_SCALEFIX равным    |
//| true.                                                            |
//+------------------------------------------------------------------+
bool ChartFixedMinSet(const double value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_POINTS_PER_BAR** - значение масштаба в пунктах на бар.

```
//+------------------------------------------------------------------+
//| Функция получает значение масштаба графика в пунктах на бар.     |
//+------------------------------------------------------------------+
double ChartPointsPerBarGet(const long chart_ID=0)
  {
//--- подготовим переменную для получения результата
   double result=EMPTY_VALUE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return(result);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает значение масштаба графика в пунктах на бар.|
//| Для того чтобы посмотреть результат изменения значения этого     |
//| свойства, нужно предварительно задать значение свойства          |
//| CHART_SCALE_PT_PER_BAR равным true.                              |
//+------------------------------------------------------------------+
bool ChartPointsPerBarSet(const double value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_PRICE_MIN** - возвращает значение минимума графика.

```
//+------------------------------------------------------------------+
//| Функция получает значение минимума графика в главном окне или    |
//| подокне.                                                         |
//+------------------------------------------------------------------+
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
  {
//--- подготовим переменную для получения результата
   double result=EMPTY_VALUE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_PRICE_MAX** - возвращает значение максимума графика.

```
//+------------------------------------------------------------------+
//| Функция получает значение максимума графика в главном окне или   |
//| подокне.                                                         |
//+------------------------------------------------------------------+
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
  {
//--- подготовим переменную для получения результата
   double result=EMPTY_VALUE;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- вернем значение свойства графика
   return(result);
  }
```

- **CHART_COMMENT** - текст комментария на графике.

```
//+------------------------------------------------------------------+
//| Функция получает текст комментария в левом верхнем углу графика. |
//+------------------------------------------------------------------+
bool ChartCommentGet(string &result,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetString(chart_ID,CHART_COMMENT,result))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция устанавливает текст комментария в левом верхнем углу     |
//| графика.                                                         |
//+------------------------------------------------------------------+
bool ChartCommentSet(const string str,const long chart_ID=0)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetString(chart_ID,CHART_COMMENT,str))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_IS_MAXIMIZED** - окно графика развернуто

```
//+------------------------------------------------------------------+
//| Функция определяет, является ли данное окно графика развернутым  |
//+------------------------------------------------------------------+
bool ChartWindowsIsMaximized(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_IS_MAXIMIZED))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
```

- **CHART_IS_MINIMIZED** - окно графика свернуто

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Функция определяет, является ли данное окно графика свернутым    |
//+------------------------------------------------------------------+
bool ChartWindowsIsMinimized(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_IS_MINIMIZED))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
```

Панель для свойств графика

```
//--- подключим библиотеку элементов управления
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- предопределенные константы
#define X_PROPERTY_NAME_1     10  // x-координата имени свойства в первом столбце
#define X_PROPERTY_VALUE_1    225 // x-координата значения свойства в первом столбце
#define X_PROPERTY_NAME_2     345 // x-координата имени свойства во втором и третьем ст
#define X_PROPERTY_VALUE_2    550 // x-координата значения свойства во втором и третьем
#define X_BUTTON_1            285 // x-координата кнопки в первом столбце
#define X_BUTTON_2            700 // x-координата кнопки во втором столбце
#define Y_PROPERTY_1          30  // y-координата начала первого и второго столбца
#define Y_PROPERTY_2          286 // y-координата начала третьего столбца
#define Y_DISTANCE            16  // расстояние по оси y между строками
#define LAST_PROPERTY_NUMBER 111 // номер последнего графического свойства
//--- входные параметры
input color InpFirstColor=clrDodgerBlue; // Цвет нечетных строк
input color InpSecondColor=clrGoldenrod; // Цвет четных строк
//--- переменные и массивы
CChartObjectLabel  ExtLabelsName[];  // надписи для отображения имен свойств
CChartObjectLabel  ExtLabelsValue[]; // надписи для отображения значений свойств
CChartObjectButton ExtButtons[];     // кнопки
int                ExtNumbers[];     // индексы свойств
string             ExtNames[];       // имена свойств
uchar              ExtDataTypes[];   // типы данных свойств (integer, double, string)
uint               ExtGroupTypes[];  // массив, который хранит данные о принадлежности
uchar              ExtDrawTypes[];   // массив, который хранит данные о способе отобра
double             ExtMaxValue[];    // максимальные значения свойств, которые они мо
double             ExtMinValue[];    // минимальные значения свойств, которые они могу
```

© 2000-2025, MetaQuotes Ltd.

```
double            ExtStep[];        // шаги для изменений свойств
int               ExtCount;         // общее количество всех свойств
color             ExtColors[2];     // массив цветов для отображения строк
string            ExtComments[2];   // массив комментариев (для свойства CHART_COMMEN
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- выведем комментарий на график
   Comment("SomeComment");
//--- сохраним цвета в массив для дальнейшего переключения между ними
   ExtColors[0]=InpFirstColor;
   ExtColors[1]=InpSecondColor;
//--- сохраним комментарии в массив для дальнейшего переключения между ними
   ExtComments[0]="FirstComment";
   ExtComments[1]="SecondComment";
//--- подготовим и отобразим панель управления свойствами графика
   if(!PrepareControls())
      return(INIT_FAILED);
//--- успешное выполнение
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Deinitialization function of the expert                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- очищаем текст комментария на графике
   Comment("");
  }
//+------------------------------------------------------------------+
//| Обработчик событий графика                                       |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- проверка события нажатия на объект графика
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      //--- разобьем имя объекта по разделителю
      string obj_name[];
      StringSplit(sparam,'_',obj_name);
      //--- проверка, является ли объект кнопкой
      if(obj_name[0]=="Button")
        {
         //--- получаем индекс кнопки
```

© 2000-2025, MetaQuotes Ltd.

```
         int index=(int)StringToInteger(obj_name[1]);
         //--- установим кнопку в ненажатое состояние
         ExtButtons[index].State(false);
         //--- установим новое значения свойства в зависимости от его типа
         if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
         if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
         if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
        }
     }
//--- перерисовка значений свойств
   RedrawProperties();
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Изменение целочисленного свойства графика                        |
//+------------------------------------------------------------------+
void ChangeIntegerProperty(const int index)
  {
//--- получаем текущее значение свойства
   long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
//--- определяем следующее значение свойства
   switch(ExtDrawTypes[index])
     {
      case 'C':
         value=GetNextColor((color)value);
         break;
      default:
         value=(long)GetNextValue((double)value,index);
         break;
     }
//--- устанавливаем новое значение свойства
   ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
  }
//+------------------------------------------------------------------+
//| Изменение вещественного свойства графика                         |
//+------------------------------------------------------------------+
void ChangeDoubleProperty(const int index)
  {
//--- получаем текущее значение свойства
   double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
//--- определяем следующее значение свойства
   value=GetNextValue(value,index);
//--- устанавливаем новое значение свойства
   ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Изменение строкового свойства графика                            |
//+------------------------------------------------------------------+
void ChangeStringProperty(const int index)
  {
//--- статическая переменная для переключения внутри массива комментариев ExtComments
   static uint comment_index=1;
//--- меняем индекс для получения другого комментария
   comment_index=1-comment_index;
//--- устанавливаем новое значение свойства
   ChartSetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[index],ExtComments[comment_
  }
//+------------------------------------------------------------------+
//| Определение следующего значения свойства                         |
//+------------------------------------------------------------------+
double GetNextValue(const double value,const int index)
  {
   if(value+ExtStep[index]<=ExtMaxValue[index])
      return(value+ExtStep[index]);
   else
      return(ExtMinValue[index]);
  }
//+------------------------------------------------------------------+
//| Получение следующего цвета для свойства типа color               |
//+------------------------------------------------------------------+
color GetNextColor(const color clr)
  {
//--- вернем следующее значение цвета
   switch(clr)
     {
      case clrWhite: return(clrRed);
      case clrRed:   return(clrGreen);
      case clrGreen: return(clrBlue);
      case clrBlue:  return(clrBlack);
      default:       return(clrWhite);
     }
  }
//+------------------------------------------------------------------+
//| Перерисовка значений свойств                                     |
//+------------------------------------------------------------------+
void RedrawProperties(void)
  {
//--- текст значения свойства
   string text;
   long   value;
//--- цикл по количеству свойств
   for(int i=0;i<ExtCount;i++)
     {
      text="";
      switch(ExtDataTypes[i])
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          case 'I':
             //--- получаем текущее значение свойства
             if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[i],0,value))
             break;
             //--- текст целочисленного свойства
             switch(ExtDrawTypes[i])
               {
                //--- свойство цвета
                case 'C':
                   text=(string)((color)value);
                   break;
                   //--- булевое свойство
                case 'B':
                   text=(string)((bool)value);
                   break;
                   //--- свойство перечисления ENUM_CHART_MODE
                case 'M':
                   text=EnumToString((ENUM_CHART_MODE)value);
                   break;
                   //--- свойство перечисления ENUM_CHART_VOLUME_MODE
                case 'V':
                   text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
                   break;
                   //--- число типа int
                default:
                   text=IntegerToString(value);
                   break;
               }
             break;
          case 'D':
             //--- текст вещественного свойства
             text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumber
             break;
          case 'S':
             //--- текст строкового свойства
             text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
             break;
         }
       //--- отобразим значение свойства
       ExtLabelsValue[i].Description(text);
      }
   }
//+------------------------------------------------------------------+
//| Создание панели для управления свойствами графика                |
//+------------------------------------------------------------------+
bool PrepareControls(void)
   {
//--- выделим память под массивы с запасом
```

© 2000-2025, MetaQuotes Ltd.

```
   MemoryAllocation(LAST_PROPERTY_NUMBER+1);
//--- переменные
   int i=0;      // переменная цикла
   int col_1=0; // количество свойств в первом столбце
   int col_2=0; // количество свойств во втором столбце
   int col_3=0; // количество свойств в третьем столбце
//--- текущее количество свойств - 0
   ExtCount=0;
//--- ищем свойства в цикле
   while(i<=LAST_PROPERTY_NUMBER)
     {
      //--- запомним текущий номер свойства
      ExtNumbers[ExtCount]=i;
      //--- увеличим значение переменной цикла
      i++;
      //--- проверим, есть ли свойство с таким номером
      if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],Ex
        {
         //--- создаем элементы управления для свойства
         switch(ExtGroupTypes[ExtCount])
           {
            case 1:
               //--- создаем надписи и кнопку для свойства
               if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BUTT
               return(false);
               //--- количество элементов в первом столбце увеличилось
               col_1++;
               break;
            case 2:
               //--- создаем надписи и кнопку для свойства
               if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BUTT
               return(false);
               //--- количество элементов во втором столбце увеличилось
               col_2++;
               break;
            case 3:
               //--- создаем только надписи для свойства
               if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,Y_PR
               return(false);
               //--- количество элементов в третьем столбце увеличилось
               col_3++;
               break;
           }
         //--- определим максимальное, минимальное значение свойства и шаг
         GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCount
         //--- увеличим количество свойств
         ExtCount++;
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- освободим память, которая не используется массивами
   MemoryAllocation(ExtCount);
//--- перерисуем значения свойств
   RedrawProperties();
   ChartRedraw();
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Выделение памяти для массивов                                    |
//+------------------------------------------------------------------+
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);
  }
//+------------------------------------------------------------------+
//| Проверяем индекс свойства на принадлежность к одному из          |
//| перечислений ENUM_CHART_PROPERTIES                               |
//+------------------------------------------------------------------+
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar &d
  {
//--- проверяем, является ли свойство целочисленным
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
   if(_LastError==0)
     {
      data_type='I';                       // свойство из перечисления ENUM_CHART_PROPE
      GetTypes(ind,group_type,draw_type); // определим параметры отображения свойства
      return(true);
     }
//--- проверяем, является ли свойство вещественным
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
   if(_LastError==0)
     {
      data_type='D';                       // свойство из перечисления ENUM_CHART_PROPE
      GetTypes(ind,group_type,draw_type); // определим параметры отображения свойства
      return(true);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- проверяем, является ли свойство строковым
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
   if(_LastError==0)
     {
      data_type='S';                       // свойство из перечисления ENUM_CHART_PROPE
      GetTypes(ind,group_type,draw_type); // определим параметры отображения свойства
      return(true);
     }
//--- свойство не принадлежит ни одному перечислению
   return(false);
  }
//+------------------------------------------------------------------+
//| Определение того, в какой группе должно находиться свойство,     |
//| и его тип отображения                                            |
//+------------------------------------------------------------------+
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- проверим на принадлежность свойства к третьей группе
//--- свойства третьей группы отображаются во втором столбце, начиная с CHART_BRING_TO
   if(CheckThirdGroup(property_number,group_type,draw_type))
      return;
//--- проверим на принадлежность свойства ко второй группе
//--- свойства второй группы отображаются во втором столбце с начала
   if(CheckSecondGroup(property_number,group_type,draw_type))
      return;
//--- если попали сюда, значит свойство принадлежит первой группе (первый столбец)
   CheckFirstGroup(property_number,group_type,draw_type);
  }
//+------------------------------------------------------------------+
//| Функция проверяет, принадлежит ли свойство третьей группе, и     |
//| если да, то определяет его тип отображения                       |
//+------------------------------------------------------------------+
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- проверим свойство на принадлежность к третьей группе
   switch(property_number)
     {
      //--- булевые свойства
      case CHART_IS_OBJECT:
      case CHART_WINDOW_IS_VISIBLE:
         draw_type='B';
         break;
         //--- целочисленные свойства
      case CHART_VISIBLE_BARS:
      case CHART_WINDOWS_TOTAL:
      case CHART_WINDOW_HANDLE:
      case CHART_WINDOW_YDISTANCE:
      case CHART_FIRST_VISIBLE_BAR:
```

© 2000-2025, MetaQuotes Ltd.

```
      case CHART_WIDTH_IN_BARS:
      case CHART_WIDTH_IN_PIXELS:
         draw_type='I';
         break;
         //--- вещественные свойства
      case CHART_PRICE_MIN:
      case CHART_PRICE_MAX:
         draw_type='D';
         break;
         //--- по сути, это свойство является командой показа графика поверх всех дру
         //--- для данной панели в его применении нет необходимости, так как окно все
         //--- будет становиться поверх всех других раньше, чем мы используем его
      case CHART_BRING_TO_TOP:
         draw_type=' ';
         break;
         //--- свойство не принадлежит к третьей группе
      default:
         return(false);
     }
//--- свойство принадлежит к третьей группе
   group_type=3;
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция проверяет, принадлежит ли свойство второй группе, и если |
//| да, то определяет его тип отображения                            |
//+------------------------------------------------------------------+
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- проверим свойство на принадлежность ко второй группе
   switch(property_number)
     {
      //--- свойство типа ENUM_CHART_MODE
      case CHART_MODE:
         draw_type='M';
         break;
         //--- свойство типа ENUM_CHART_VOLUME_MODE
      case CHART_SHOW_VOLUMES:
         draw_type='V';
         break;
         //--- строковое свойство
      case CHART_COMMENT:
         draw_type='S';
         break;
         //--- свойство цвета
      case CHART_COLOR_BACKGROUND:
      case CHART_COLOR_FOREGROUND:
      case CHART_COLOR_GRID:
      case CHART_COLOR_VOLUME:
```

© 2000-2025, 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;
         //--- свойство не принадлежит ко второй группе
      default:
         return(false);
     }
//--- свойство принадлежит ко второй группе
   group_type=2;
   return(true);
  }
//+------------------------------------------------------------------+
//| Эта функция вызывается только в том случае, если уже известно,   |
//| что свойство не принадлежит второй и третьей группам свойств     |
//+------------------------------------------------------------------+
void CheckFirstGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- свойство принадлежит первой группе
   group_type=1;
//--- определим тип отображения свойства
   switch(property_number)
     {
      //--- целочисленные свойства
      case CHART_SCALE:
      case CHART_HEIGHT_IN_PIXELS:
         draw_type='I';
         return;
         //--- вещественные свойства
      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;
         //--- остались только булевые свойства
      default:
         draw_type='B';
         return;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Создание надписи и кнопки для свойства                           |
//+------------------------------------------------------------------+
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
  {
//--- статический массив для переключения внутри массива цвета ExtColors
   static uint color_index[3]={1,1,1};
//--- меняем индекс для получения другого цвета
   color_index[type]=1-color_index[type];
//--- выведем надписи и кнопку (если btn=true) для свойства
   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);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Создание надписи                                                 |
//+------------------------------------------------------------------+
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);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Создание кнопки                                                  |
//+------------------------------------------------------------------+
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("Next"))
      return(false);
   if(!btn.FontSize(10))
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!btn.Color(clrBlack))
      return(false);
   if(!btn.BackColor(clrWhite))
      return(false);
   if(!btn.BorderColor(clrBlack))
      return(false);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Устанавливаем максимальное, минимальное значение свойства и шаг  |
//+------------------------------------------------------------------+
void GetMaxMinStep(const int property_number,double &max,double &min,double &step)
  {
   double value;
//--- установим значения в зависимости от типа свойства
   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;
```

© 2000-2025, MetaQuotes Ltd.

```
         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;
         step=30;
         break;
         //--- значения по умолчанию
      default:
         max=1;
         min=0;
         step=1;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Константы объектов

Предусмотрено 44 графических объекта, которые можно создавать и отображать на ценовом графике. Все константы для работы с объектами разбиты на 9 групп:

- Типы объектов - идентификаторы графических объектов;
- Свойства объектов - работа со свойствами графических объектов;
- Способы привязки объектов - константы позиционирования объектов на графике;
- Угол привязки - позволяет установить угла графика, относительно которого производится позиционирование объекта в пикселях;
- Видимость объектов - задание таймфреймов, на которых объект является видимым;
- Уровни волн Эллиотта - градация волновой разметки;
- Объекты Ганна - константы тренда для веера Ганна и сетки Ганна;
- Набор Web-цветов - константы предопределенных Web-цветов;
- Wingdings - коды символов шрифта Wingdings.

© 2000-2025, MetaQuotes Ltd.

Типы объектов

При создании графического объекта функцией ObjectCreate() необходимо указать тип создаваемого объекта, который может принимать одно из значений перечисления ENUM_OBJECT. Дальнейшие уточнения свойств созданного объекта возможно с помощью функций по работе с графическими объектами.

ENUM_OBJECT

Идентификатор		Описание
OBJ_VLINE		Вертикальная линия
OBJ_HLINE		Горизонтальная линия
OBJ_TREND		Трендовая линия
OBJ_TRENDBYANGLE		Трендовая линия по углу
OBJ_CYCLES		Циклические линии
OBJ_ARROWED_LINE		Объект "Линия со стрелкой"
OBJ_CHANNEL		Равноудаленный канал
OBJ_STDDEVCHANNEL		Канал стандартного отклонения
OBJ_REGRESSION		Канал на линейной регрессии
OBJ_PITCHFORK		Вилы Эндрюса
OBJ_GANNLINE		Линия Ганна
OBJ_GANNFAN		Веер Ганна
OBJ_GANNGRID		Сетка Ганна
OBJ_FIBO		Уровни Фибоначчи
OBJ_FIBOTIMES		Временные зоны Фибоначчи
OBJ_FIBOFAN		Веер Фибоначчи
OBJ_FIBOARC		Дуги Фибоначчи
OBJ_FIBOCHANNEL		Канал Фибоначчи
OBJ_EXPANSION		Расширение Фибоначчи
OBJ_ELLIOTWAVE5		5-волновка Эллиота
OBJ_ELLIOTWAVE3		3-волновка Эллиота
OBJ_RECTANGLE		Прямоугольник
OBJ_TRIANGLE		Треугольник
OBJ_ELLIPSE		Эллипс
OBJ_ARROW_THUMB_UP		Знак "Хорошо" (большой палец вверх)

© 2000-2025, MetaQuotes Ltd.

Идентификатор		Описание
OBJ_ARROW_THUMB_DOWN		Знак "Плохо" (большой палец вниз)
OBJ_ARROW_UP		Знак "Стрелка вверх"
OBJ_ARROW_DOWN		Знак "Стрелка вниз"
OBJ_ARROW_STOP		Знак "Стоп"
OBJ_ARROW_CHECK		Знак "Птичка" (галка)
OBJ_ARROW_LEFT_PRICE		Левая ценовая метка
OBJ_ARROW_RIGHT_PRICE		Правая ценовая метка
OBJ_ARROW_BUY		Знак "Buy"
OBJ_ARROW_SELL		Знак "Sell"
OBJ_ARROW		Объект "Стрелка"
OBJ_TEXT		Объект "Текст"
OBJ_LABEL		Объект "Текстовая метка"
OBJ_BUTTON		Объект "Кнопка"
OBJ_CHART		Объект "График"
OBJ_BITMAP		Объект "Рисунок"
OBJ_BITMAP_LABEL		Объект "Графическая метка"
OBJ_EDIT		Объект "Поле ввода"
OBJ_EVENT		Объект "Событие", соответствующий событию в экономическом календаре
OBJ_RECTANGLE_LABEL		Объект "Прямоугольная метка"для создания и оформления пользовательского графического интерфейса.

© 2000-2025, MetaQuotes Ltd.

OBJ_VLINE

Вертикальная линия.



Примечание

При создании вертикальной линии, можно указать режим отображения линии на все окна графика (свойство OBJPROP_RAY).

Пример

Следующий скрипт создает и перемещает на графике вертикальную линию. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Вертикальная линия\"."
#property description "Дата точки привязки задается в процентах от ширины"
#property description "окна графика в барах."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="VLine";      // Имя линии
input int              InpDate=25;           // Дата линии в %
input color            InpColor=clrRed;      // Цвет линии
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH;  // Стиль линии
input int              InpWidth=3;           // Толщина линии
input bool             InpBack=false;        // Линия на заднем плане
input bool             InpSelection=true;    // Выделить для перемещений
input bool             InpRay=true;          // Продолжение линии вниз
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpHidden=true;      // Скрыт в списке объектов
input long            InpZOrder=0;         // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает вертикальную линию                                       |
//+------------------------------------------------------------------+
bool VLineCreate(const long            chart_ID=0,        // ID графика
                 const string          name="VLine",      // имя линии
                 const int             sub_window=0,      // номер подокна
                 datetime              time=0,            // время линии
                 const color           clr=clrRed,        // цвет линии
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии
                 const int             width=1,           // толщина линии
                 const bool            back=false,        // на заднем плане
                 const bool            selection=true,    // выделить для перемещений
                 const bool            ray=true,          // продолжение линии вниз
                 const bool            hidden=true,       // скрыт в списке объектов
                 const long            z_order=0)         // приоритет на нажатие мыш
  {
//--- если время линии не задано, то проводим ее через последний бар
   if(!time)
      time=TimeCurrent();
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим вертикальную линию
   if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать вертикальную линию! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет линии
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль отображения линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения линии мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим отображения линии в подокнах графика
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещение вертикальной линии                                   |
//+------------------------------------------------------------------+
bool VLineMove(const long   chart_ID=0,   // ID графика
               const string name="VLine", // имя линии
               datetime     time=0)       // время линии
  {
//--- если время линии не задано, то перемещаем ее на последний бар
   if(!time)
      time=TimeCurrent();
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим вертикальную линию
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": не удалось переместить вертикальную линию! Код ошибки = ",GetLastError
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет вертикальную линию                                       |
//+------------------------------------------------------------------+
bool VLineDelete(const long   chart_ID=0,   // ID графика
                 const string name="VLine") // имя линии
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим вертикальную линию
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить вертикальную линию! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- массив для хранения значений дат, которые будут использованы
//--- для установки и изменения координаты точки привязки линии
   datetime date[];
//--- выделение памяти
   ArrayResize(date,bars);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- определим точки для рисования линии
   int d=InpDate*(bars-1)/100;
//--- создадим вертикальную линию
   if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpRay,InpHidden,InpZOrder))
      return;
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать линию
//--- счетчик цикла
   int h_steps=bars/2;
//--- перемещаем линию
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d<bars-1)
         d+=1;
      //--- сдвигаем точку
      if(!VLineMove(0,InpName,date[d]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.03 секунды
      Sleep(30);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим канал с графика
   VLineDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_HLINE

Горизонтальная линия.



Пример

Следующий скрипт создает и перемещает на графике горизонтальную линию. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Горизонтальная линия\"."
#property description "Цена точки привязки задается в процентах от высоты"
#property description "окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="HLine";      // Имя линии
input int              InpPrice=25;          // Цена линии в %
input color            InpColor=clrRed;      // Цвет линии
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH;  // Стиль линии
input int              InpWidth=3;           // Толщина линии
input bool             InpBack=false;        // Линия на заднем плане
input bool             InpSelection=true;    // Выделить для перемещений
input bool             InpHidden=true;       // Скрыт в списке объектов
input long             InpZOrder=0;          // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает горизонтальную линию                                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool HLineCreate(const long            chart_ID=0,        // ID графика
                 const string          name="HLine",      // имя линии
                 const int             sub_window=0,      // номер подокна
                 double                price=0,           // цена линии
                 const color           clr=clrRed,        // цвет линии
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии
                 const int             width=1,           // толщина линии
                 const bool            back=false,        // на заднем плане
                 const bool            selection=true,    // выделить для перемещений
                 const bool            hidden=true,       // скрыт в списке объектов
                 const long            z_order=0)         // приоритет на нажатие мыши
  {
//--- если цена не задана, то установим ее на уровне текущей цены Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим горизонтальную линию
   if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать горизонтальную линию! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет линии
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль отображения линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения линии мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещение горизонтальной линии                                 |
//+------------------------------------------------------------------+
bool HLineMove(const long   chart_ID=0,   // ID графика
```

© 2000-2025, MetaQuotes Ltd.

```
                const string name="HLine", // имя линии
                double       price=0)      // цена линии
  {
//--- если цена линии не задана, то перемещаем ее на уровень текущей цены Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим горизонтальную линию
   if(!ObjectMove(chart_ID,name,0,0,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить горизонтальную линию! Код ошибки = ",GetLastErro
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет горизонтальную линию                                     |
//+------------------------------------------------------------------+
bool HLineDelete(const long   chart_ID=0,   // ID графика
                 const string name="HLine") // имя линии
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим горизонтальную линию
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить горизонтальную линию! Код ошибки = ",GetLastError())
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- размер массива price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- массив для хранения значений цен, которые будут использованы
//--- для установки и изменения координаты точки привязки линии
   double price[];
//--- выделение памяти
   ArrayResize(price,accuracy);
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования линии
   int p=InpPrice*(accuracy-1)/100;
//--- создадим горизонтальную линию
   if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать линию
//--- счетчик цикла
   int v_steps=accuracy/2;
//--- перемещаем линию
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p<accuracy-1)
         p+=1;
      //--- сдвигаем точку
      if(!HLineMove(0,InpName,price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим с графика
   HLineDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TREND

Трендовая линия.



Примечание

Для трендовой линии можно указать режим продолжения ее отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно).

Пример

Следующий скрипт создает и перемещает на графике трендовую линию. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Трендовая линия\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="Trend";      // Имя линии
input int             InpDate1=35;          // Дата 1-ой точки в %
input int             InpPrice1=60;         // Цена 1-ой точки в %
input int             InpDate2=65;          // Дата 2-ой точки в %
input int             InpPrice2=40;         // Цена 2-ой точки в %
input color           InpColor=clrRed;      // Цвет линии
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Стиль линии
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpWidth=2;          // Толщина линии
input bool             InpBack=false;       // Линия на заднем плане
input bool             InpSelection=true;   // Выделить для перемещений
input bool             InpRayLeft=false;    // Продолжение линии влево
input bool             InpRayRight=false;   // Продолжение линии вправо
input bool             InpHidden=true;      // Скрыт в списке объектов
input long             InpZOrder=0;         // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает линию тренда по заданным координатам                     |
//+------------------------------------------------------------------+
bool TrendCreate(const long            chart_ID=0,        // ID графика
                 const string          name="TrendLine",  // имя линии
                 const int             sub_window=0,      // номер подокна
                 datetime              time1=0,           // время первой точки
                 double                price1=0,          // цена первой точки
                 datetime              time2=0,           // время второй точки
                 double                price2=0,          // цена второй точки
                 const color           clr=clrRed,        // цвет линии
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии
                 const int             width=1,           // толщина линии
                 const bool            back=false,        // на заднем плане
                 const bool            selection=true,    // выделить для перемещений
                 const bool            ray_left=false,    // продолжение линии влево
                 const bool            ray_right=false,   // продолжение линии вправо
                 const bool            hidden=true,       // скрыт в списке объектов
                 const long            z_order=0)         // приоритет на нажатие мыш
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeTrendEmptyPoints(time1,price1,time2,price2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим трендовую линию по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": не удалось создать линию тренда! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет линии
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль отображения линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения линии мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
```

© 2000-2025, MetaQuotes Ltd.

```
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения линии влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения линии вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки линии тренда                           |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID графика
                      const string name="TrendLine", // имя линии
                      const int    point_index=0,    // номер точки привязки
                      datetime     time=0,           // координата времени точки привя
                      double       price=0)          // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки линии тренда
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция удаляет линию тренда с графика.                          |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // ID графика
                 const string name="TrendLine") // имя линии
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим линию тренда
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить линию тренда! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки линии тренда и для пустых      |
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- если время первой точки не задано, то она будет на текущем баре
   if(!time1)
      time1=TimeCurrent();
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время второй точки не задано, то она лежит на 9 баров левее второй
   if(!time2)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- установим вторую точку на 9 баров левее первой
      time2=temp[0];
     }
//--- если цена второй точки не задана, то она совпадает с ценой первой точки
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки линии
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования линии
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- создадим линию тренда
   if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,InpStyle
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки линии
//--- счетчик цикла
   int v_steps=accuracy/5;
//--- перемещаем первую точку привязки по вертикали
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1>1)
         p1-=1;
      //--- сдвигаем точку
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
         return;
       //--- перерисуем график
       ChartRedraw();
      }
//--- перемещаем вторую точку привязки по вертикали
   for(int i=0;i<v_steps;i++)
     {
       //--- возьмем следующее значение
       if(p2<accuracy-1)
         p2+=1;
       //--- сдвигаем точку
       if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
         return;
       //--- перерисуем график
       ChartRedraw();
      }
//--- задержка в полсекунды
   Sleep(500);
//--- счетчик цикла
   int h_steps=bars/2;
//--- перемещаем обе точки привязки по горизонтали одновременно
   for(int i=0;i<h_steps;i++)
     {
       //--- возьмем следующие значения
       if(d1<bars-1)
         d1+=1;
       if(d2>1)
         d2-=1;
       //--- сдвигаем точки
       if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
       if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
         return;
       //--- перерисуем график
       ChartRedraw();
       // задержка в 0.03 секунды
       Sleep(30);
      }
//--- задержка в 1 секунду
   Sleep(1000);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- удалим трендовую линию
   TrendDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TRENDBYANGLE

Трендовая линия по углу.



Примечание

Для трендовой линии можно указать режим продолжения ее отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно).

Для установки наклона линии можно использовать как угол, так и координаты второй точки привязки.

Пример

Следующий скрипт создает и перемещает на графике трендовую линию. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Трендовая линия по углу\"."
#property description "Координаты точек привязки задаются в процентах от размеров"
#property description "окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="Trend";      // Имя линии
input int              InpDate1=50;          // Дата 1-ой точки в %
input int              InpPrice1=75;         // Цена 1-ой точки в %
input int              InpAngle=0;           // Угол наклона линии
input color            InpColor=clrRed;      // Цвет линии
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Стиль линии
input int             InpWidth=2;          // Толщина линии
input bool            InpBack=false;       // Линия на заднем плане
input bool            InpSelection=true;   // Выделить для перемещений
input bool            InpRayLeft=false;    // Продолжение линии влево
input bool            InpRayRight=true;    // Продолжение линии вправо
input bool            InpHidden=true;      // Скрыт в списке объектов
input long            InpZOrder=0;         // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает линию тренда по углу                                     |
//+------------------------------------------------------------------+
bool TrendByAngleCreate(const long            chart_ID=0,        // ID графика
                        const string          name="TrendLine",  // имя линии
                        const int             sub_window=0,      // номер подокна
                        datetime              time=0,            // время точки
                        double                price=0,           // цена точки
                        const double          angle=45.0,        // угол наклона
                        const color           clr=clrRed,        // цвет линии
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии
                        const int             width=1,           // толщина линии
                        const bool            back=false,        // на заднем плане
                        const bool            selection=true,    // выделить для перем
                        const bool            ray_left=false,    // продолжение линии
                        const bool            ray_right=true,    // продолжение линии
                        const bool            hidden=true,       // скрыт в списке об
                        const long            z_order=0)         // приоритет на нажат
  {
//--- для того, чтобы было удобно перемещать трендовую линию мышью, создадим вторую то
   datetime time2=0;
   double   price2=0;
//--- установим координаты точек привязки, если они не заданы
   ChangeTrendEmptyPoints(time,price,time2,price2);
//--- сбросим значение ошибки
   ResetLastError();
//--- строим трендовую линию по 2-ум точкам
   if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price2)
     {
      Print(__FUNCTION__,
            ": не удалось создать линию тренда! Код ошибки = ",GetLastError());
      return(false);
     }
//--- изменяем угол наклона трендовой линии; в процессе изменения угла, координата вто
//--- точки линии переопределится автоматически в соответствии с новым значением угла
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- установим цвет линии
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения линии мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения линии влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения линии вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет координаты точки привязки линии тренда                  |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID графика
                      const string name="TrendLine", // имя линии
                      datetime     time=0,           // координата времени точки привя
                      double       price=0)          // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки линии тренда
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет угол наклона линии тренда                               |
//+------------------------------------------------------------------+
bool TrendAngleChange(const long   chart_ID=0,       // ID графика
```

© 2000-2025, MetaQuotes Ltd.

```
                      const string name="TrendLine", // имя линии тренда
                      const double angle=45)         // угол наклона линии тренда
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим угол наклона линии тренда
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": не удалось изменить угол наклона линии! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет линию тренда                                             |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,        // ID графика
                 const string name="TrendLine") // имя линии
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим линию тренда
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить линию тренда! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки линии тренда и для пустых      |
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- если время первой точки не задано, то она будет на текущем баре
   if(!time1)
      time1=TimeCurrent();
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- установим координаты второй, вспомогательной точки
//--- вторая точка будет лежать левее на 9 баров и иметь ту же цену
   datetime second_point_time[10];
```

© 2000-2025, MetaQuotes Ltd.

```
   CopyTime(Symbol(),Period(),time1,10,second_point_time);
   time2=second_point_time[0];
   price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки линии
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования линии
   int d1=InpDate1*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- создадим линию тренда
   if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать и вращать линию
//--- счетчик цикла
   int v_steps=accuracy/2;
//--- перемещаем точку привязки и изменяем угол наклона линии
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1>1)
         p1-=1;
      //--- сдвигаем точку
      if(!TrendPointChange(0,InpName,date[d1],price[p1]))
         return;
      if(!TrendAngleChange(0,InpName,18*(i+1)))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим с графика
   TrendDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_CYCLES

Циклические линии.



Примечание

Расстояние между линиями задается координатами времени двух точек привязки объекта.

Пример

Следующий скрипт создает и перемещает на графике циклические линии. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит циклические линии на графике."
#property description "Координаты точек привязки задаются в процентах"
#property description "от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="Cycles";   // Имя объекта
input int             InpDate1=10;        // Дата 1-ой точки в %
input int             InpPrice1=45;       // Цена 1-ой точки в %
input int             InpDate2=20;        // Дата 2-ой точки в %
input int             InpPrice2=55;       // Цена 2-ой точки в %
input color           InpColor=clrRed;    // Цвет циклических линий
input ENUM_LINE_STYLE InpStyle=STYLE_DOT; // Стиль циклических линий
input int             InpWidth=1;         // Толщина циклических линий
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpBack=false;       // Объект на заднем плане
input bool            InpSelection=true;   // Выделить для перемещений
input bool            InpHidden=true;      // Скрыт в списке объектов
input long            InpZOrder=0;         // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает циклические линии                                        |
//+------------------------------------------------------------------+
bool CyclesCreate(const long            chart_ID=0,         // ID графика
                  const string          name="Cycles",      // имя объекта
                  const int             sub_window=0,       // номер подокна
                  datetime              time1=0,            // время первой точки
                  double                price1=0,           // цена первой точки
                  datetime              time2=0,            // время второй точки
                  double                price2=0,           // цена второй точки
                  const color           clr=clrRed,         // цвет циклических линий
                  const ENUM_LINE_STYLE style=STYLE_SOLID,  // стиль циклических линий
                  const int             width=1,            // толщина циклических лини
                  const bool            back=false,         // на заднем плане
                  const bool            selection=true,     // выделить для перемещений
                  const bool            hidden=true,        // скрыт в списке объектов
                  const long            z_order=0)          // приоритет на нажатие мы
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeCyclesEmptyPoints(time1,price1,time2,price2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим циклические линии по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": не удалось создать циклические линии! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет линий
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль отображения линий
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения линий мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool CyclesPointChange(const long   chart_ID=0,     // ID графика
                       const string name="Cycles", // имя объекта
                       const int    point_index=0, // номер точки привязки
                       datetime     time=0,        // координата времени точки привязк
                       double       price=0)       // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет циклические линии                                        |
//+------------------------------------------------------------------+
bool CyclesDelete(const long   chart_ID=0,     // ID графика
                  const string name="Cycles") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим циклические линии
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить циклические линии! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки циклических линий и для пустых |
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
  {
//--- если время первой точки не задано, то она будет на текущем баре
   if(!time1)
      time1=TimeCurrent();
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время второй точки не задано, то она лежит на 9 баров левее второй
   if(!time2)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- установим вторую точку на 9 баров левее первой
      time2=temp[0];
     }
//--- если цена второй точки не задана, то она совпадает с ценой первой точки
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки циклических линий
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования циклических линий
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- создадим линию тренда
   if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки
//--- счетчик цикла
   int h_steps=bars/5;
//--- перемещаем вторую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d2<bars-1)
         d2+=1;
      //--- сдвигаем точку
      if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   h_steps=bars/4;
//--- перемещаем первую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d1<bars-1)
         d1+=1;
      //--- сдвигаем точку
      if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим объект с графика
   CyclesDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROWED_LINE

Линия со стрелкой.



Пример

Следующий скрипт создает и перемещает на графике линию со стрелкой. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Линия со стрелкой\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="ArrowedLine"; // Имя линии
input int             InpDate1=35;           // Дата 1-ой точки в %
input int             InpPrice1=60;          // Цена 1-ой точки в %
input int             InpDate2=65;           // Дата 2-ой точки в %
input int             InpPrice2=40;          // Цена 2-ой точки в %
input color           InpColor=clrRed;       // Цвет линии
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;   // Стиль линии
input int             InpWidth=2;            // Толщина линии
input bool            InpBack=false;         // Линия на заднем плане
input bool            InpSelection=true;     // Выделить для перемещений
input bool            InpHidden=true;        // Скрыт в списке объектов
input long            InpZOrder=0;           // Приоритет на нажатие мышью
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Создает линию со стрелкой по заданным координатам                |
//+------------------------------------------------------------------+
bool ArrowedLineCreate(const long            chart_ID=0,          // ID графика
                       const string          name="ArrowedLine", // имя линии
                       const int             sub_window=0,       // номер подокна
                       datetime              time1=0,            // время первой точки
                       double                price1=0,           // цена первой точки
                       datetime              time2=0,            // время второй точки
                       double                price2=0,           // цена второй точки
                       const color           clr=clrRed,         // цвет линии
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // стиль линии
                       const int             width=1,            // толщина линии
                       const bool            back=false,         // на заднем плане
                       const bool            selection=true,     // выделить для перем
                       const bool            hidden=true,        // скрыт в списке об
                       const long            z_order=0)          // приоритет на нажат
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим линию со стрелкой по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,price
     {
      Print(__FUNCTION__,
            ": не удалось создать линию со стрелкой! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет линии
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль отображения линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения линии мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки линии со стрелкой                      |
//+------------------------------------------------------------------+
bool ArrowedLinePointChange(const long   chart_ID=0,          // ID графика
                            const string name="ArrowedLine", // имя линии
                            const int    point_index=0,       // номер точки привязки
                            datetime     time=0,              // координата времени точ
                            double       price=0)             // координата цены точки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки линии
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция удаляет линию со стрелкой с графика                      |
//+------------------------------------------------------------------+
bool ArrowedLineDelete(const long   chart_ID=0,          // ID графика
                       const string name="ArrowedLine") // имя линии
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим линию со стрелкой
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить линию со стрелкой! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки линии  и для пустых значений   |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2)
  {
//--- если время первой точки не задано, то она будет на текущем баре
   if(!time1)
      time1=TimeCurrent();
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время второй точки не задано, то она лежит на 9 баров левее первой
   if(!time2)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- установим вторую точку на 9 баров левее первой
      time2=temp[0];
     }
//--- если цена второй точки не задана, то она совпадает с ценой первой точки
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки линии
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования линии
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- создадим линию со стрелкой
   if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки линии
//--- счетчик цикла
   int v_steps=accuracy/5;
//--- перемещаем вторую точку привязки по вертикали
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p2<accuracy-1)
         p2+=1;
      //--- сдвигаем точку
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- перемещаем первую точку привязки по вертикали
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1>1)
         p1-=1;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- сдвигаем точку
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в полсекунды
   Sleep(500);
//--- счетчик цикла
   int h_steps=bars/2;
//--- перемещаем обе точки привязки по горизонтали одновременно
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующие значения
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- сдвигаем точки
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.03 секунды
      Sleep(30);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим линию со стрелкой
   ArrowedLineDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_CHANNEL

Равноудаленный канал.



Примечание

Для равноудаленного канала можно указать режим продолжения его отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно). Также можно установить режим заливки канала цветом.

Пример

Следующий скрипт создает и перемещает на графике равноудаленный канал. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Равноудаленный канал\"."
#property description "Координаты точек привязки задаются в процентах от размеров"
#property description "окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="Channel";   // Имя канала
input int             InpDate1=25;         // Дата 1-ой точки в %
input int             InpPrice1=60;        // Цена 1-ой точки в %
input int             InpDate2=65;         // Дата 2-ой точки в %
input int             InpPrice2=80;        // Цена 2-ой точки в %
input int             InpDate3=30;         // Дата 3-ей точки в %
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpPrice3=40;         // Цена 3-ей точки в %
input color            InpColor=clrRed;      // Цвет канала
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH;  // Стиль линий канала
input int              InpWidth=2;           // Толщина линий канала
input bool             InpBack=false;        // Канал на заднем плане
input bool             InpFill=false;        // Заливка канала цветом
input bool             InpSelection=true;    // Выделить для перемещений
input bool             InpRayLeft=false;     // Продолжение канала влево
input bool             InpRayRight=false;    // Продолжение канала вправо
input bool             InpHidden=true;       // Скрыт в списке объектов
input long             InpZOrder=0;          // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает равноудаленный канал по заданным координатам             |
//+------------------------------------------------------------------+
bool ChannelCreate(const long            chart_ID=0,        // ID графика
                   const string          name="Channel",    // имя канала
                   const int             sub_window=0,      // номер подокна
                   datetime              time1=0,           // время первой точки
                   double                price1=0,          // цена первой точки
                   datetime              time2=0,           // время второй точки
                   double                price2=0,          // цена второй точки
                   datetime              time3=0,           // время третьей точки
                   double                price3=0,          // цена третьей точки
                   const color           clr=clrRed,        // цвет канала
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линий канала
                   const int             width=1,           // толщина линий канала
                   const bool            fill=false,        // заливка канала цветом
                   const bool            back=false,        // на заднем плане
                   const bool            selection=true,    // выделить для перемещени
                   const bool            ray_left=false,    // продолжение канала влев
                   const bool            ray_right=false,   // продолжение канала впра
                   const bool            hidden=true,       // скрыт в списке объектов
                   const long            z_order=0)         // приоритет на нажатие мы
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим канал по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,tim
     {
      Print(__FUNCTION__,
            ": не удалось создать равноудаленный канал! Код ошибки = ",GetLastError())
      return(false);
     }
//--- установим цвет канала
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий канала
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим толщину линий канала
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- включим (true) или отключим (false) режим заливки канала
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения канала для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения канала влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения канала вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки канала                                 |
//+------------------------------------------------------------------+
bool ChannelPointChange(const long   chart_ID=0,      // ID графика
                        const string name="Channel", // имя канала
                        const int    point_index=0,  // номер точки привязки
                        datetime     time=0,         // координата времени точки привя
                        double       price=0)        // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Удаляет канал                                                    |
//+------------------------------------------------------------------+
bool ChannelDelete(const long   chart_ID=0,     // ID графика
                   const string name="Channel") // имя канала
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим канал
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить канал! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки канала и для пустых значений   |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- если время второй (правой) точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если цена второй точки не задана, то она будет иметь значение Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время первой (левой) точки не задано, то она лежит на 9 баров левее второй
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то сдвинем ее на 300 пунктов выше второй
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- если время третьей точки не задано, то оно совпадает с временем первой точки
   if(!time3)
      time3=time1;
//--- если цена третьей точки не задана, то она совпадает с ценой второй точки
   if(!price3)
      price3=price2;
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   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("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки канала
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования канала
   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;
//--- создадим равноудаленный канал
   if(!ChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
```

© 2000-2025, MetaQuotes Ltd.

```
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки канала
//--- счетчик цикла
   int h_steps=bars/6;
//--- перемещаем вторую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d2<bars-1)
         d2+=1;
      //--- сдвигаем точку
      if(!ChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- перемещаем первую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d1>1)
         d1-=1;
      //--- сдвигаем точку
      if(!ChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
```

© 2000-2025, MetaQuotes Ltd.

```
   int v_steps=accuracy/10;
//--- перемещаем третью точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p3>1)
         p3-=1;
      //--- сдвигаем точку
      if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим канал с графика
   ChannelDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_STDDEVCHANNEL

Канал стандартного отклонения.



Примечание

Для канала стандартного отклонения можно указать режим продолжения его отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно). Также можно установить режим заливки канала цветом.

Для изменения значения отклонения канала используется свойство OBJPROP_DEVIATION.

Пример

Следующий скрипт создает и перемещает на графике канал стандартного отклонения. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Канал стандартного отклонени
#property description "Координаты точек привязки задаются в процентах от размеров"
#property description "окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="StdDevChannel";   // Имя канала
input int              InpDate1=10;               // Дата 1-ой точки в %
input int              InpDate2=40;               // Дата 2-ой точки в %
input double           InpDeviation=1.0;          // Отклонение
input color            InpColor=clrRed;           // Цвет канала
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Стиль линий канала
input int             InpWidth=2;                // Толщина линий канала
input bool            InpFill=false;             // Заливка канала цветом
input bool            InpBack=false;             // Канал на заднем плане
input bool            InpSelection=true;         // Выделить для перемещений
input bool            InpRayLeft=false;          // Продолжение канала влево
input bool            InpRayRight=false;         // Продолжение канала вправо
input bool            InpHidden=true;            // Скрыт в списке объектов
input long            InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает канал стандартного отклонения по заданным координатам    |
//+------------------------------------------------------------------+
bool StdDevChannelCreate(const long             chart_ID=0,        // ID графика
                         const string           name="Channel",    // имя канала
                         const int              sub_window=0,      // номер подокна
                         datetime               time1=0,           // время первой точк
                         datetime               time2=0,           // время второй точк
                         const double           deviation=1.0,     // отклонение
                         const color            clr=clrRed,        // цвет канала
                         const ENUM_LINE_STYLE  style=STYLE_SOLID, // стиль линий канал
                         const int              width=1,           // толщина линий ка
                         const bool             fill=false,        // заливка канала ц
                         const bool             back=false,        // на заднем плане
                         const bool             selection=true,    // выделить для пере
                         const bool             ray_left=false,    // продолжение канал
                         const bool             ray_right=false,   // продолжение канал
                         const bool             hidden=true,       // скрыт в списке об
                         const long             z_order=0)         // приоритет на нажа
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeChannelEmptyPoints(time1,time2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим канал по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать канал стандартного отклонения! Код ошибки = ",GetLas
      return(false);
     }
//--- установим величину отклонения, от нее зависит ширина канала
   ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- установим цвет канала
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий канала
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий канала
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- включим (true) или отключим (false) режим заливки канала
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения канала для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения канала влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения канала вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки канала                                 |
//+------------------------------------------------------------------+
bool StdDevChannelPointChange(const long   chart_ID=0,      // ID графика
                              const string name="Channel", // имя канала
                              const int    point_index=0,  // номер точки привязки
                              datetime     time=0)         // координата времени точки
  {
//--- если время точки не задано, то перемещаем ее на текущий бар
   if(!time)
      time=TimeCurrent();
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет отклонение канала                                       |
//+------------------------------------------------------------------+
bool StdDevChannelDeviationChange(const long   chart_ID=0,      // ID графика
                                  const string name="Channel", // имя канала
                                  const double deviation=1.0)  // отклонение
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим угол наклона линии тренда
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
     {
      Print(__FUNCTION__,
            ": не удалось изменить отклонение канала! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет канал                                                    |
//+------------------------------------------------------------------+
bool StdDevChannelDelete(const long   chart_ID=0,     // ID графика
                         const string name="Channel") // имя канала
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим канал
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить канал! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки канала и для пустых значений   |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
  {
//--- если время второй точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если время первой точки не задано, то она лежит на 9 баров левее второй
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим вторую точку на 9 баров левее второй
      time1=temp[0];
     }
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки канала
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования канала
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- создадим канал стандартного отклонения
   if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation,InpColor,InpStyl
      InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать канал по горизонтали вправо и расширять его
//--- счетчик цикла
   int h_steps=bars/2;
//--- перемещаем канал
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующие значения
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- переместим точки привязки
      if(!StdDevChannelPointChange(0,InpName,0,date[d1]))
         return;
      if(!StdDevChannelPointChange(0,InpName,1,date[d2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   double v_steps=InpDeviation*2;
//--- расширим канал
   for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
     {
      if(!StdDevChannelDeviationChange(0,InpName,i))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим канал с графика
   StdDevChannelDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

OBJ_REGRESSION

Канал на линейной регрессии.



Примечание

Для канала на линейной регрессии можно указать режим продолжения его отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно). Также можно установить режим заливки канала цветом.

Пример

Следующий скрипт создает и перемещает на графике канал на линейной регрессии. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Канал на линейной регрессии\
#property description "Координаты точек привязки задаются в процентах от размеров"
#property description "окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="Regression"; // Имя канала
input int             InpDate1=10;          // Дата 1-ой точки в %
input int             InpDate2=40;          // Дата 2-ой точки в %
input color           InpColor=clrRed;      // Цвет канала
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Стиль линий канала
input int             InpWidth=2;           // Толщина линий канала
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpFill=false;         // Заливка канала цветом
input bool              InpBack=false;         // Канал на заднем плане
input bool              InpSelection=true;     // Выделить для перемещений
input bool              InpRayLeft=false;      // Продолжение канала влево
input bool              InpRayRight=false;     // Продолжение канала вправо
input bool              InpHidden=true;        // Скрыт в списке объектов
input long              InpZOrder=0;           // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает канал на линейной регрессии по заданным координатам      |
//+------------------------------------------------------------------+
bool RegressionCreate(const long             chart_ID=0,         // ID графика
                      const string           name="Regression", // имя канала
                      const int              sub_window=0,       // номер подокна
                      datetime               time1=0,            // время первой точки
                      datetime               time2=0,            // время второй точки
                      const color            clr=clrRed,         // цвет канала
                      const ENUM_LINE_STYLE  style=STYLE_SOLID, // стиль линий канала
                      const int              width=1,            // толщина линий канала
                      const bool             fill=false,         // заливка канала цвето
                      const bool             back=false,         // на заднем плане
                      const bool             selection=true,     // выделить для перемещ
                      const bool             ray_left=false,     // продолжение канала в
                      const bool             ray_right=false,    // продолжение канала в
                      const bool             hidden=true,        // скрыт в списке объек
                      const long             z_order=0)          // приоритет на нажатие
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeRegressionEmptyPoints(time1,time2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим канал по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать канал на линейной регрессии! Код ошибки = ",GetLastE
      return(false);
     }
//--- установим цвет канала
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий канала
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий канала
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- включим (true) или отключим (false) режим заливки канала
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения канала для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
```

© 2000-2025, MetaQuotes Ltd.

```
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения канала влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения канала вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки канала                                 |
//+------------------------------------------------------------------+
bool RegressionPointChange(const long   chart_ID=0,     // ID графика
                           const string name="Channel", // имя канала
                           const int    point_index=0,  // номер точки привязки
                           datetime     time=0)         // координата времени точки пр
  {
//--- если время точки не задано, то перемещаем ее на текущий бар
   if(!time)
      time=TimeCurrent();
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет канал                                                    |
//+------------------------------------------------------------------+
bool RegressionDelete(const long   chart_ID=0,     // ID графика
                      const string name="Channel") // имя канала
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим канал
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": не удалось удалить канал! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки канала и для пустых значений   |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
  {
//--- если время второй точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если время первой точки не задано, то она лежит на 9 баров левее второй
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки канала
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования канала
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- создадим канал на линейной регрессии
   if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
      InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать канал по горизонтали вправо
//--- счетчик цикла
   int h_steps=bars/2;
//--- перемещаем канал
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующие значения
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- переместим точки привязки
      if(!RegressionPointChange(0,InpName,0,date[d1]))
         return;
      if(!RegressionPointChange(0,InpName,1,date[d2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
```

© 2000-2025, MetaQuotes Ltd.

```
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим канал с графика
   RegressionDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_PITCHFORK

Вилы Эндрюса.



Примечание

Для "Вил Эндрюса" можно указать режим продолжения отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно).

Также можно указать количество линий-уровней, их значения и цвет.

Пример

Следующий скрипт создает и перемещает на графике "Вилы Эндрюса". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Вилы Эндрюса\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="Pitchfork";  // Имя вил
input int             InpDate1=14;          // Дата 1-ой точки в %
input int             InpPrice1=40;         // Цена 1-ой точки в %
input int             InpDate2=18;          // Дата 2-ой точки в %
input int             InpPrice2=50;         // Цена 2-ой точки в %
input int             InpDate3=18;          // Дата 3-ей точки в %
input int             InpPrice3=30;         // Цена 3-ей точки в %
```

© 2000-2025, MetaQuotes Ltd.

```
input color           InpColor=clrRed;      // Цвет вил
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Стиль линий вил
input int             InpWidth=1;           // Толщина линий вил
input bool            InpBack=false;        // Вилы на заднем плане
input bool            InpSelection=true;    // Выделить для перемещений
input bool            InpRayLeft=false;     // Продолжение вил влево
input bool            InpRayRight=false;    // Продолжение вил вправо
input bool            InpHidden=true;       // Скрыт в списке объектов
input long            InpZOrder=0;          // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Вилы Эндрюса" по заданным координатам                   |
//+------------------------------------------------------------------+
bool PitchforkCreate(const long            chart_ID=0,        // ID графика
                     const string          name="Pitchfork",  // имя вил
                     const int             sub_window=0,      // номер подокна
                     datetime              time1=0,           // время первой точки
                     double                price1=0,          // цена первой точки
                     datetime              time2=0,           // время второй точки
                     double                price2=0,          // цена второй точки
                     datetime              time3=0,           // время третьей точки
                     double                price3=0,          // цена третьей точки
                     const color           clr=clrRed,        // цвет линий вил
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линий вил
                     const int             width=1,           // толщина линий вил
                     const bool            back=false,        // на заднем плане
                     const bool            selection=true,    // выделить для перемеще
                     const bool            ray_left=false,    // продолжение вил влево
                     const bool            ray_right=false,   // продолжение вил впра
                     const bool            hidden=true,       // скрыт в списке объект
                     const long            z_order=0)         // приоритет на нажатие
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Вилы Эндрюса" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Вилы Эндрюса\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения вил для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения вил влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения вил вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Задает количество уровней "Вил Эндрюса" и их параметры           |
//+------------------------------------------------------------------+
bool PitchforkLevelsSet(int             levels,           // количество линий уровня
                        double          &values[],        // значения линий уровня
                        color           &colors[],        // цвет линий уровня
                        ENUM_LINE_STYLE &styles[],        // стиль линий уровня
                        int             &widths[],        // толщина линий уровня
                        const long      chart_ID=0,       // ID графика
                        const string    name="Pitchfork") // имя вил
  {
//--- проверим размеры массивов
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": длина массива не соответствуют количеству уровней, ошибка!
      return(false);
     }
//--- установим количество уровней
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- установим свойства уровней в цикле
   for(int i=0;i<levels;i++)
     {
      //--- значение уровня
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- цвет уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- стиль уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- толщина уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- описание уровня
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Вил Эндрюса"                          |
//+------------------------------------------------------------------+
bool PitchforkPointChange(const long   chart_ID=0,       // ID графика
                          const string name="Pitchfork", // имя канала
                          const int    point_index=0,    // номер точки привязки
                          datetime     time=0,           // координата времени точки п
                          double       price=0)          // координата цены точки прив
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет "Вилы Эндрюса"                                           |
//+------------------------------------------------------------------+
bool PitchforkDelete(const long   chart_ID=0,       // ID графика
                     const string name="Pitchfork") // имя канала
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим канал
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Вилы Эндрюса\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Вил Эндрюса" и для пустых     |
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- если время второй (правой верхней) точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если цена второй точки не задана, то она будет иметь значение Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время первой (левой) точки не задано, то она лежит на 9 баров левее второй
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то сдвинем ее на 200 пунктов ниже второй
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- если время третьей точки не задано, то оно совпадает с временем второй точки
   if(!time3)
      time3=time2;
//--- если цена третьей точки не задана, то сдвинем ее на 200 пунктов ниже первой
   if(!price3)
      price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   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("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
```

© 2000-2025, MetaQuotes Ltd.

```
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки "Вил Эндрюса"
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования "Вил Эндрюса"
   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;
//--- создадим вилы
   if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pric
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки вил
//--- счетчик цикла
   int v_steps=accuracy/10;
//--- перемещаем первую точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1>1)
         p1-=1;
      //--- сдвигаем точку
```

© 2000-2025, MetaQuotes Ltd.

```
      if(!PitchforkPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int h_steps=bars/8;
//--- перемещаем третью точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d3<bars-1)
         d3+=1;
      //--- сдвигаем точку
      if(!PitchforkPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   v_steps=accuracy/10;
//--- перемещаем вторую точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p2>1)
         p2-=1;
      //--- сдвигаем точку
      if(!PitchforkPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим вилы с графика
   PitchforkDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNLINE

Линия Ганна.



Примечание

Для "Линии Ганна" можно указать режим продолжения ее отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно).

Для установки наклона линии можно использовать как угол Ганна с масштабом, так и координаты второй точки привязки.

Пример

Следующий скрипт создает и перемещает на графике "Линию Ганна". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Линия Ганна\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="GannLine";         // Имя линии
input int              InpDate1=20;                // Дата 1-ой точки в %
input int              InpPrice1=75;               // Цена 1-ой точки в %
input int              InpDate2=80;                // Дата 2-ой точки в %
input double           InpAngle=0.0;               // Угол Ганна
```

© 2000-2025, MetaQuotes Ltd.

```
input double           InpScale=1.0;               // Масштаб
input color            InpColor=clrRed;            // Цвет линии
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT;  // Стиль линии
input int              InpWidth=2;                 // Толщина линии
input bool             InpBack=false;              // Линия на заднем плане
input bool             InpSelection=true;          // Выделить для перемещений
input bool             InpRayLeft=false;           // Продолжение линии влево
input bool             InpRayRight=true;           // Продолжение линии вправо
input bool             InpHidden=true;             // Скрыт в списке объектов
input long             InpZOrder=0;                // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Линию Ганна" по координатам, углу и масштабу            |
//+------------------------------------------------------------------+
bool GannLineCreate(const long             chart_ID=0,         // ID графика
                    const string           name="GannLine",    // имя линии
                    const int              sub_window=0,       // номер подокна
                    datetime               time1=0,            // время первой точки
                    double                 price1=0,           // цена первой точки
                    datetime               time2=0,            // время второй точки
                    const double           angle=1.0,          // угол Ганна
                    const double           scale=1.0,          // масштаб
                    const color            clr=clrRed,         // цвет линии
                    const ENUM_LINE_STYLE  style=STYLE_SOLID,  // стиль линии
                    const int              width=1,            // толщина линии
                    const bool             back=false,         // на заднем плане
                    const bool             selection=true,     // выделить для перемещен
                    const bool             ray_left=false,     // продолжение линии влев
                    const bool             ray_right=true,     // продолжение линии впра
                    const bool             hidden=true,        // скрыт в списке объекто
                    const long             z_order=0)          // приоритет на нажатие м
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeGannLineEmptyPoints(time1,price1,time2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Линию Ганна" по заданным координатам
//--- правильная координата цены второй точки привязки переопределится
//--- автоматически после изменения угла Ганна и (или) масштаба,
   if(!ObjectCreate(chart_ID,name,OBJ_GANNLINE,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Линию Ганна\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- изменим угол Ганна
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- изменим масштаб (количество пипсов на бар)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- установим цвет линии
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль отображения линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения линии для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения линии влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения линии вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Линии Ганна"                          |
//+------------------------------------------------------------------+
bool GannLinePointChange(const long   chart_ID=0,       // ID графика
                         const string name="GannLine", // имя линии
                         const int    point_index=0,   // номер точки привязки
                         datetime     time=0,          // координата времени точки при
                         double       price=0)         // координата цены точки привяз
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки линии
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Изменяет угол Ганна                                              |
//+------------------------------------------------------------------+
bool GannLineAngleChange(const long   chart_ID=0,      // ID графика
                         const string name="GannLine", // имя линии
                         const double angle=1.0)       // угол Ганна
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим угол Ганна
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": не удалось изменить угол Ганна! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет масштаб "Линии Ганна"                                   |
//+------------------------------------------------------------------+
bool GannLineScaleChange(const long   chart_ID=0,      // ID графика
                         const string name="GannLine", // имя линии
                         const double scale=1.0)       // масштаб
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим масштаб (количество пипсов на бар)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": не удалось изменить масштаб! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция удаляет "Линию Ганна" с графика                          |
//+------------------------------------------------------------------+
bool GannLineDelete(const long   chart_ID=0,      // ID графика
                    const string name="GannLine") // имя линии
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим линию Ганна
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": не удалось удалить \"Линию Ганна\"! Код ошибки = ",GetLastError());
       return(false);
      }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Линии Ганна" и для пустых     |
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- если время второй точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если время первой точки не задано, то она лежит на 9 баров левее первой
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки линии
   datetime date[];
```

© 2000-2025, MetaQuotes Ltd.

```
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования линии Ганна
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- создадим линию Ганна
   if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle,InpScale,InpCol
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки линии и менять угол
//--- счетчик цикла
   int v_steps=accuracy/2;
//--- перемещаем первую точку привязки по вертикали
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1>1)
         p1-=1;
      //--- сдвигаем точку
      if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- задержка в полсекунды
   Sleep(500);
//--- определим текущее значение угла Ганна (изменился
//--- после перемещения первой точки привязки)
   double curr_angle;
   if(!ObjectGetDouble(0,InpName,OBJPROP_ANGLE,0,curr_angle))
      return;
//--- счетчик цикла
   v_steps=accuracy/8;
//--- изменяем угол Ганна
   for(int i=0;i<v_steps;i++)
     {
      if(!GannLineAngleChange(0,InpName,curr_angle-0.05*i))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим линию с графика
   GannLineDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNFAN

Веер Ганна.



Примечание

Для "Веера Ганна" можно указать тип тренда из перечисления ENUM_GANN_DIRECTION. Регулируя значение масштаба (OBJPROP_SCALE), можно менять угол наклона линий веера.

Пример

Следующий скрипт создает и перемещает на графике "Веер Ганна". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Веер Ганна\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="GannFan";          // Имя веера
input int             InpDate1=15;                // Дата 1-ой точки в %
input int             InpPrice1=25;               // Цена 1-ой точки в %
input int             InpDate2=85;                // Дата 2-ой точки в %
input double          InpScale=2.0;               // Масштаб
input bool            InpDirection=false;         // Направление тренда
input color           InpColor=clrRed;            // Цвет веера
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Стиль линий веера
input int             InpWidth=1;                // Толщина линий веера
input bool            InpBack=false;             // Веер на заднем плане
input bool            InpSelection=true;         // Выделить для перемещений
input bool            InpHidden=true;            // Скрыт в списке объектов
input long            InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Веер Ганна"                                             |
//+------------------------------------------------------------------+
bool GannFanCreate(const long            chart_ID=0,        // ID графика
                   const string          name="GannFan",    // имя веера
                   const int             sub_window=0,      // номер подокна
                   datetime              time1=0,           // время первой точки
                   double                price1=0,          // цена первой точки
                   datetime              time2=0,           // время второй точки
                   const double          scale=1.0,         // масштаб
                   const bool            direction=true,    // направление тренда
                   const color           clr=clrRed,        // цвет веера
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линий веера
                   const int             width=1,           // толщина линий веера
                   const bool            back=false,        // на заднем плане
                   const bool            selection=true,    // выделить для перемещени
                   const bool            hidden=true,       // скрыт в списке объектов
                   const long            z_order=0)         // приоритет на нажатие мы
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeGannFanEmptyPoints(time1,price1,time2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Веер Ганна" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Веер Ганна\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- изменим масштаб (количество пипсов на бар)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- изменим направление тренда "Веера Ганна" (true - нисходящий, false - восходящий)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- установим цвет веера
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль отображения линий веера
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий веера
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения веера для перемещений
```

© 2000-2025, MetaQuotes Ltd.

```
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Веера Ганна"                          |
//+------------------------------------------------------------------+
bool GannFanPointChange(const long   chart_ID=0,     // ID графика
                        const string name="GannFan", // имя веера
                        const int    point_index=0,  // номер точки привязки
                        datetime     time=0,         // координата времени точки привя
                        double       price=0)        // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки веера
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет масштаб "Веера Ганна"                                   |
//+------------------------------------------------------------------+
bool GannFanScaleChange(const long   chart_ID=0,     // ID графика
                        const string name="GannFan", // имя веера
                        const double scale=1.0)      // масштаб
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим масштаб (количество пипсов на бар)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": не удалось изменить масштаб! Код ошибки = ",GetLastError());
       return(false);
      }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет направление тренда "Веера Ганна"                        |
//+------------------------------------------------------------------+
bool GannFanDirectionChange(const long   chart_ID=0,      // ID графика
                            const string name="GannFan", // имя веера
                            const bool   direction=true) // направление тренда
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим направление тренда "Веера Ганна"
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": не удалось изменить направление тренда! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция удаляет "Веер Ганна" с графика                           |
//+------------------------------------------------------------------+
bool GannFanDelete(const long   chart_ID=0,      // ID графика
                   const string name="GannFan") // имя веера
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим веер Ганна
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Веер Ганна\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Веера Ганна" и для пустых     |
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- если время второй точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если время первой точки не задано, то она лежит на 9 баров левее второй
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки веера
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
```

© 2000-2025, MetaQuotes Ltd.

```
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования веера Ганна
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- создадим веер Ганна
   if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки веера
//--- счетчик цикла
   int v_steps=accuracy/2;
//--- перемещаем первую точку привязки по вертикали
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1<accuracy-1)
         p1+=1;
      //--- сдвигаем точку
      if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- изменим направление тренда веера на нисходящее
   GannFanDirectionChange(0,InpName,true);
//--- перерисуем график
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим веер с графика
   GannFanDelete(0,InpName);
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNGRID

Сетка Ганна.



Примечание

Для "Сетки Ганна" можно указать тип тренда из перечисления ENUM_GANN_DIRECTION. Регулируя значение масштаба (OBJPROP_SCALE), можно менять угол наклона линий сетки.

Пример

Следующий скрипт создает и перемещает на графике "Сетку Ганна". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Сетка Ганна\"."
#property description "Координаты точек привязки сетки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="GannGrid";        // Имя сетки
input int             InpDate1=15;               // Дата 1-ой точки в %
input int             InpPrice1=25;              // Цена 1-ой точки в %
input int             InpDate2=35;               // Дата 2-ой точки в %
input double          InpScale=3.0;              // Масштаб
input bool            InpDirection=false;        // Направление тренда
input color           InpColor=clrRed;           // Цвет сетки
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Стиль линий сетки
input int             InpWidth=1;                // Толщина линий веера
input bool            InpBack=false;             // Сетка на заднем плане
input bool            InpSelection=true;         // Выделить для перемещений
input bool            InpHidden=true;            // Скрыт в списке объектов
input long            InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Сетку Ганна"                                            |
//+------------------------------------------------------------------+
bool GannGridCreate(const long            chart_ID=0,        // ID графика
                    const string          name="GannGrid",   // имя сетки
                    const int             sub_window=0,      // номер подокна
                    datetime              time1=0,           // время первой точки
                    double                price1=0,          // цена первой точки
                    datetime              time2=0,           // время второй точки
                    const double          scale=1.0,         // масштаб
                    const bool            direction=true,    // направление тренда
                    const color           clr=clrRed,        // цвет сетки
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линий сетки
                    const int             width=1,           // толщина линий сетки
                    const bool            back=false,        // на заднем плане
                    const bool            selection=true,    // выделить для перемещен
                    const bool            hidden=true,       // скрыт в списке объекто
                    const long            z_order=0)         // приоритет на нажатие м
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeGannGridEmptyPoints(time1,price1,time2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Сетка Ганна" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Сетку Ганна\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- изменим масштаб (количество пипсов на бар)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- изменим направление тренда "Сетки Ганна" (true - нисходящий, false - восходящий)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- установим цвет сетки
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль отображения линий сетки
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий сетки
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения сетки для перемещений
```

© 2000-2025, MetaQuotes Ltd.

```
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Сетки Ганна"                          |
//+------------------------------------------------------------------+
bool GannGridPointChange(const long   chart_ID=0,      // ID графика
                         const string name="GannGrid", // имя объекта "Сетка Ганна"
                         const int    point_index=0,   // номер точки привязки
                         datetime     time=0,          // координата времени точки при
                         double       price=0)         // координата цены точки привяз
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки сетки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет масштаб "Сетки Ганна"                                   |
//+------------------------------------------------------------------+
bool GannGridScaleChange(const long   chart_ID=0,      // ID графика
                         const string name="GannGrid", // имя сетки
                         const double scale=1.0)       // масштаб
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим масштаб (количество пипсов на бар)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": не удалось изменить масштаб! Код ошибки = ",GetLastError());
       return(false);
      }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет направление тренда "Сетки Ганна"                        |
//+------------------------------------------------------------------+
bool GannGridDirectionChange(const long   chart_ID=0,       // ID графика
                             const string name="GannGrid", // имя сетки
                             const bool   direction=true)  // направление тренда
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим направление тренда "Сетки Ганна"
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": не удалось изменить направление тренда! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция удаляет "Сетку Ганна" с графика                          |
//+------------------------------------------------------------------+
bool GannGridDelete(const long   chart_ID=0,       // ID графика
                    const string name="GannGrid") // имя сетки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим сетку Ганна
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Сетку Ганна\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Сетки Ганна" и для пустых     |
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- если время второй точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если время первой точки не задано, то она лежит на 9 баров левее первой
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки сетки
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
```

© 2000-2025, MetaQuotes Ltd.

```
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования сетки Ганна
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- создадим сетку Ганна
   if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки сетки
//--- счетчик цикла
   int v_steps=accuracy/4;
//--- перемещаем первую точку привязки по вертикали
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1<accuracy-1)
         p1+=1;
      if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int h_steps=bars/4;
//--- перемещаем вторую точку привязки по горизонтали
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d2<bars-1)
         d2+=1;
      if(!GannGridPointChange(0,InpName,1,date[d2],0))
         return;
```

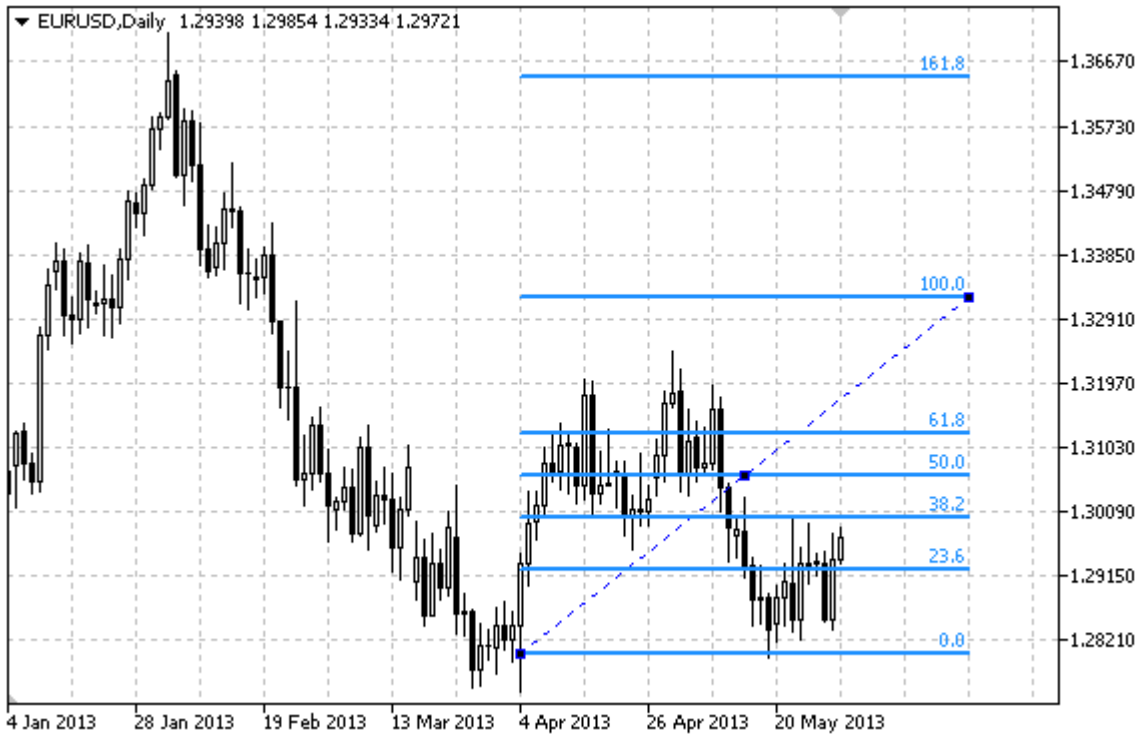
© 2000-2025, MetaQuotes Ltd.

```
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- изменим направление тренда сетки на нисходящее
   GannGridDirectionChange(0,InpName,true);
//--- перерисуем график
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим сетку с графика
   GannGridDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBO

Уровни Фибоначчи.



Примечание

Для "Уровней Фибоначчи" можно указать режим продолжения отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно).

Также можно указать количество линий-уровней, их значения и цвет.

Пример

Следующий скрипт создает и перемещает на графике "Уровни Фибоначчи". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Уровни Фибоначчи\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="FiboLevels";      // Имя объекта
input int              InpDate1=10;               // Дата 1-ой точки в %
input int              InpPrice1=65;              // Цена 1-ой точки в %
input int              InpDate2=90;               // Дата 2-ой точки в %
input int              InpPrice2=85;              // Цена 2-ой точки в %
input color            InpColor=clrRed;           // Цвет объекта
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Стиль линии
input int             InpWidth=2;                // Толщина линии
input bool            InpBack=false;             // Объект на заднем плане
input bool            InpSelection=true;         // Выделить для перемещений
input bool            InpRayLeft=false;          // Продолжение объекта влево
input bool            InpRayRight=false;         // Продолжение объекта вправо
input bool            InpHidden=true;            // Скрыт в списке объектов
input long            InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Уровни Фибоначчи" по заданным координатам               |
//+------------------------------------------------------------------+
bool FiboLevelsCreate(const long            chart_ID=0,        // ID графика
                      const string          name="FiboLevels", // имя объекта
                      const int             sub_window=0,      // номер подокна
                      datetime              time1=0,           // время первой точки
                      double                price1=0,          // цена первой точки
                      datetime              time2=0,           // время второй точки
                      double                price2=0,          // цена второй точки
                      const color           clr=clrRed,        // цвет объекта
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии объекта
                      const int             width=1,           // толщина линии объект
                      const bool            back=false,        // на заднем плане
                      const bool            selection=true,    // выделить для перемещ
                      const bool            ray_left=false,    // продолжение объекта
                      const bool            ray_right=false,   // продолжение объекта
                      const bool            hidden=true,       // скрыт в списке объек
                      const long            z_order=0)         // приоритет на нажати
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Уровни Фибоначчи" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Уровни Фибоначчи\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения объекта для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
```

© 2000-2025, MetaQuotes Ltd.

```
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения объекта влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения объекта вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установи приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Задает количество уровней и их параметры                         |
//+------------------------------------------------------------------+
bool FiboLevelsSet(int             levels,            // количество линий уровня
                   double          &values[],         // значения линий уровня
                   color           &colors[],         // цвет линий уровня
                   ENUM_LINE_STYLE &styles[],         // стиль линий уровня
                   int             &widths[],         // толщина линий уровня
                   const long      chart_ID=0,        // ID графика
                   const string    name="FiboLevels") // имя объекта
  {
//--- проверим размеры массивов
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) || levels!=ArraySize(widt
     {
      Print(__FUNCTION__,": длина массива не соответствует количеству уровней, ошибка!
      return(false);
     }
//--- установим количество уровней
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- установим свойства уровней в цикле
   for(int i=0;i<levels;i++)
     {
      //--- значение уровня
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- цвет уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- стиль уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- толщина уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- описание уровня
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- успешное выполнение
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Уровней Фибоначчи"                    |
//+------------------------------------------------------------------+
bool FiboLevelsPointChange(const long   chart_ID=0,        // ID графика
                           const string name="FiboLevels", // имя объекта
                           const int    point_index=0,     // номер точки привязки
                           datetime     time=0,            // координата времени точки
                           double       price=0)           // координата цены точки пр
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет "Уровни Фибоначчи"                                       |
//+------------------------------------------------------------------+
bool FiboLevelsDelete(const long   chart_ID=0,        // ID графика
                      const string name="FiboLevels") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим объект
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Уровни Фибоначчи\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Уровней Фибоначчи" и для      |
//| пустых значений устанавливает значения по умолчанию              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                 datetime &time2,double &price2)
  {
//--- если время второй точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если цена второй точки не задана, то она будет иметь значение Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время первой точки не задано, то она лежит на 9 баров левее второй
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то сдвинем ее на 200 пунктов ниже второй
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки "Уровней Фибоначчи"
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
       return;
      }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования "Уровней Фибоначчи"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- создадим объект
   if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
      {
       return;
      }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки
//--- счетчик цикла
   int v_steps=accuracy*2/5;
//--- перемещаем первую точку привязки
   for(int i=0;i<v_steps;i++)
      {
       //--- возьмем следующее значение
       if(p1>1)
          p1-=1;
       //--- сдвигаем точку
       if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
          return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
          return;
       //--- перерисуем график
       ChartRedraw();
      }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   v_steps=accuracy*4/5;
//--- перемещаем вторую точку привязки
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p2>1)
         p2-=1;
      //--- сдвигаем точку
      if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим объект с графика
   FiboLevelsDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOTIMES

Временные зоны Фибоначчи.



Примечание

Для "Временных зон Фибоначчи" можно указать количество линий-уровней, их значения и цвет.

Пример

Следующий скрипт создает и перемещает на графике "Временные зоны Фибоначчи". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Временные зоны Фибоначчи\"."
#property description "Координаты точек привязки задаются в процентах от размеров"
#property description "окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="FiboTimes";       // Имя объекта
input int             InpDate1=10;               // Дата 1-ой точки в %
input int             InpPrice1=45;              // Цена 1-ой точки в %
input int             InpDate2=20;               // Дата 2-ой точки в %
input int             InpPrice2=55;              // Цена 2-ой точки в %
input color           InpColor=clrRed;           // Цвет объекта
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Стиль линии
input int             InpWidth=2;                // Толщина линии
input bool            InpBack=false;             // Объект на заднем плане
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpSelection=true;         // Выделить для перемещений
input bool            InpHidden=true;            // Скрыт в списке объектов
input long            InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Временные зоны Фибоначчи" по заданным координатам       |
//+------------------------------------------------------------------+
bool FiboTimesCreate(const long            chart_ID=0,        // ID графика
                     const string          name="FiboTimes",  // имя объекта
                     const int             sub_window=0,      // номер подокна
                     datetime              time1=0,           // время первой точки
                     double                price1=0,          // цена первой точки
                     datetime              time2=0,           // время второй точки
                     double                price2=0,          // цена второй точки
                     const color           clr=clrRed,        // цвет объекта
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии объекта
                     const int             width=1,           // толщина линии объекта
                     const bool            back=false,        // на заднем плане
                     const bool            selection=true,    // выделить для перемеще
                     const bool            hidden=true,       // скрыт в списке объект
                     const long            z_order=0)         // приоритет на нажатие
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Временные зоны Фибоначчи" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Временные зоны Фибоначчи\"! Код ошибки = ",GetLast
      return(false);
     }
//--- установим цвет
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения объекта для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Задает количество уровней и их параметры                         |
//+------------------------------------------------------------------+
bool FiboTimesLevelsSet(int             levels,           // количество линий уровня
                        double          &values[],        // значения линий уровня
                        color           &colors[],        // цвет линий уровня
                        ENUM_LINE_STYLE &styles[],        // стиль линий уровня
                        int             &widths[],        // толщина линий уровня
                        const long      chart_ID=0,       // ID графика
                        const string    name="FiboTimes") // имя объекта
  {
//--- проверим размеры массивов
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": длина массива не соответствует количеству уровней, ошибка!
      return(false);
     }
//--- установим количество уровней
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- установим свойства уровней в цикле
   for(int i=0;i<levels;i++)
     {
      //--- значение уровня
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- цвет уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- стиль уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- толщина уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- описание уровня
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1));
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Временных зон Фибоначчи"              |
//+------------------------------------------------------------------+
bool FiboTimesPointChange(const long   chart_ID=0,       // ID графика
                          const string name="FiboTimes", // имя объекта
                          const int    point_index=0,    // номер точки привязки
                          datetime     time=0,           // координата времени точки
                          double       price=0)          // координата цены точки при
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет "Временные зоны Фибоначчи"                               |
//+------------------------------------------------------------------+
bool FiboTimesDelete(const long   chart_ID=0,        // ID графика
                     const string name="FiboTimes") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим объект
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Временные зоны Фибоначчи\"! Код ошибки = ",GetLast
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Временных зон Фибоначчи"  и   |
//| для пустых значений устанавливает значения по умолчанию          |
//+------------------------------------------------------------------+
void ChangeFiboTimesEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- если время первой точки не задано, то она будет на текущем баре
   if(!time1)
      time1=TimeCurrent();
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если время второй точки не задано, то она лежит на 2 бара левее второй
   if(!time2)
     {
      //--- массив для приема времени открытия 3 последних баров
      datetime temp[3];
      CopyTime(Symbol(),Period(),time1,3,temp);
      //--- установим первую точку на 2 бара левее второй
      time2=temp[0];
     }
//--- если цена второй точки не задана, то она совпадает с ценой первой точки
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки "Временных зон Фибоначчи"
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования "Временных зон Фибоначчи"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- создадим объект
   if(!FiboTimesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки
//--- счетчик цикла
   int h_steps=bars*2/5;
//--- перемещаем вторую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d2<bars-1)
         d2+=1;
      //--- сдвигаем точку
      if(!FiboTimesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   h_steps=bars*3/5;
//--- перемещаем первую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d1<bars-1)
         d1+=1;
      //--- сдвигаем точку
      if(!FiboTimesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим объект с графика
   FiboTimesDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOFAN

Веер Фибоначчи.



Примечание

Для "Веера Фибоначчи" можно указать количество линий-уровней, их значения и цвет.

Пример

Следующий скрипт создает и перемещает на графике "Веер Фибоначчи". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Веер Фибоначчи\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="FiboFan";          // Имя веера
input int             InpDate1=10;                // Дата 1-ой точки в %
input int             InpPrice1=25;               // Цена 1-ой точки в %
input int             InpDate2=30;                // Дата 2-ой точки в %
input int             InpPrice2=50;               // Цена 2-ой точки в %
input color           InpColor=clrRed;            // Цвет линии веера
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Стиль линии
input int             InpWidth=2;                 // Толщина линии
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpBack=false;             // Объект на заднем плане
input bool            InpSelection=true;         // Выделить для перемещений
input bool            InpHidden=true;            // Скрыт в списке объектов
input long            InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Веер Фибоначчи" по заданным координатам                 |
//+------------------------------------------------------------------+
bool FiboFanCreate(const long            chart_ID=0,        // ID графика
                   const string          name="FiboFan",    // имя веера
                   const int             sub_window=0,      // номер подокна
                   datetime              time1=0,           // время первой точки
                   double                price1=0,          // цена первой точки
                   datetime              time2=0,           // время второй точки
                   double                price2=0,          // цена второй точки
                   const color           clr=clrRed,        // цвет линии веера
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии веера
                   const int             width=1,           // толщина линии веера
                   const bool            back=false,        // на заднем плане
                   const bool            selection=true,    // выделить для перемещени
                   const bool            hidden=true,       // скрыт в списке объектов
                   const long            z_order=0)         // приоритет на нажатие мы
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Веер Фибоначчи" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Веер Фибоначчи\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения веера для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Задает количество уровней и их параметры                         |
//+------------------------------------------------------------------+
bool FiboFanLevelsSet(int             levels,         // количество линий уровня
                      double          &values[],      // значения линий уровня
                      color           &colors[],      // цвет линий уровня
                      ENUM_LINE_STYLE &styles[],      // стиль линий уровня
                      int             &widths[],      // толщина линий уровня
                      const long      chart_ID=0,     // ID графика
                      const string    name="FiboFan") // имя веера
  {
//--- проверим размеры массивов
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": длина массива не соответствует количеству уровней, ошибка!
      return(false);
     }
//--- установим количество уровней
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- установим свойства уровней в цикле
   for(int i=0;i<levels;i++)
     {
      //--- значение уровня
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- цвет уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- стиль уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- толщина уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- описание уровня
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Веера Фибоначчи"                      |
//+------------------------------------------------------------------+
bool FiboFanPointChange(const long   chart_ID=0,     // ID графика
                        const string name="FiboFan", // имя веера
                        const int    point_index=0,  // номер точки привязки
                        datetime     time=0,         // координата времени точки привя
```

© 2000-2025, MetaQuotes Ltd.

```
                        double       price=0)        // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет "Веер Фибоначчи"                                         |
//+------------------------------------------------------------------+
bool FiboFanDelete(const long   chart_ID=0,     // ID графика
                   const string name="FiboFan") // имя веера
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим веер
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Веер Фибоначчи\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Веера Фибоначчи" и для        |
//| пустых значений устанавливает значения по умолчанию              |
//+------------------------------------------------------------------+
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- если время второй точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если цена второй точки не задана, то она будет иметь значение Bid
   if(!price2)
```

© 2000-2025, MetaQuotes Ltd.

```
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время первой точки не задано, то она лежит на 9 баров левее второй
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то сдвинем ее на 200 пунктов ниже второй
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки "Веера Фибоначчи"
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
```

© 2000-2025, MetaQuotes Ltd.

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования "Веера Фибоначчи"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- создадим объект
   if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки веера
//--- счетчик цикла
   int v_steps=accuracy/2;
//--- перемещаем первую точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1<accuracy-1)
         p1+=1;
      //--- сдвигаем точку
      if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int h_steps=bars/4;
//--- перемещаем вторую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d2<bars-1)
         d2+=1;
      //--- сдвигаем точку
      if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
```

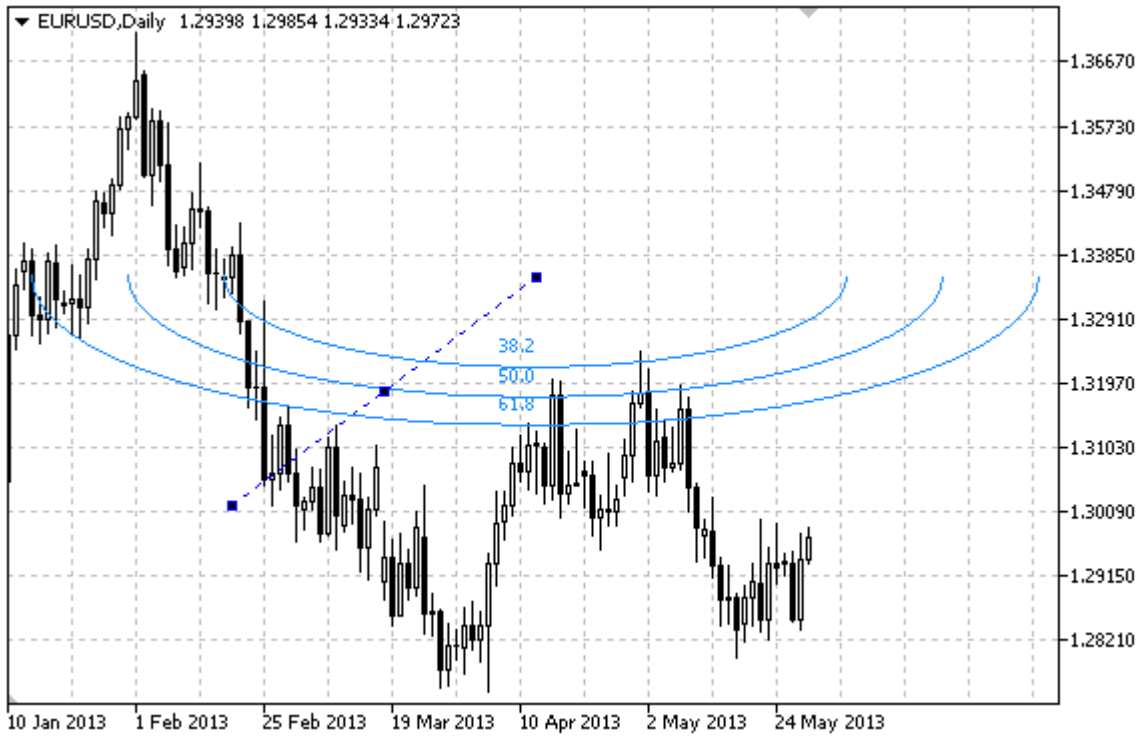
© 2000-2025, MetaQuotes Ltd.

```
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим объект с графика
   FiboFanDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOARC

Дуги Фибоначчи.



Примечание

Для "Дуг Фибоначчи" можно указать режим отображения всего эллипса целиком. Радиус кривизны линий можно задать изменяя масштаб и координаты точек привязки.

Также можно указать количество линий-уровней, их значения и цвет.

Пример

Следующий скрипт создает и перемещает на графике "Дуги Фибоначчи". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Дуги Фибоначчи\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="FiboArc";         // Имя объекта
input int             InpDate1=25;               // Дата 1-ой точки в %
input int             InpPrice1=25;              // Цена 1-ой точки в %
input int             InpDate2=35;               // Дата 2-ой точки в %
input int             InpPrice2=55;              // Цена 2-ой точки в %
input double          InpScale=3.0;              // Масштаб
```

© 2000-2025, MetaQuotes Ltd.

```
input bool             InpFullEllipse=true;       // Форма дуг
input color            InpColor=clrRed;           // Цвет линии
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Стиль линии
input int              InpWidth=2;                // Толщина линии
input bool             InpBack=false;             // Объект на заднем плане
input bool             InpSelection=true;         // Выделить для перемещений
input bool             InpHidden=true;            // Скрыт в списке объектов
input long             InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Дуги Фибоначчи" по заданным координатам                 |
//+------------------------------------------------------------------+
bool FiboArcCreate(const long            chart_ID=0,          // ID графика
                   const string          name="FiboArc",      // имя объекта
                   const int             sub_window=0,        // номер подокна
                   datetime              time1=0,             // время первой точки
                   double                price1=0,            // цена первой точки
                   datetime              time2=0,             // время второй точки
                   double                price2=0,            // цена второй точки
                   const double          scale=1.0,           // масштаб
                   const bool            full_ellipse=false,  // форма дуг
                   const color           clr=clrRed,          // цвет линии
                   const ENUM_LINE_STYLE style=STYLE_SOLID,   // стиль линии
                   const int             width=1,             // толщина линии
                   const bool            back=false,          // на заднем плане
                   const bool            selection=true,      // выделить для перемещен
                   const bool            hidden=true,         // скрыт в списке объекто
                   const long            z_order=0)           // приоритет на нажатие м
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Дуги Фибоначчи" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Дуги Фибоначчи\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим масштаб
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- установим отображение дуг в виде полного эллипса (true) или половины (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
//--- установим цвет
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линии
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения дуг для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Задает количество уровней и их параметры                         |
//+------------------------------------------------------------------+
bool FiboArcLevelsSet(int             levels,         // количество линий уровня
                      double          &values[],      // значения линий уровня
                      color           &colors[],      // цвет линий уровня
                      ENUM_LINE_STYLE &styles[],      // стиль линий уровня
                      int             &widths[],      // толщина линий уровня
                      const long      chart_ID=0,     // ID графика
                      const string    name="FiboArc") // имя объекта
  {
//--- проверим размеры массивов
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": длина массива не соответствует количеству уровней, ошибка!
      return(false);
     }
//--- установим количество уровней
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- установим свойства уровней в цикле
   for(int i=0;i<levels;i++)
     {
      //--- значение уровня
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- цвет уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- стиль уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- толщина уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- описание уровня
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Дуг Фибоначчи"                        |
//+------------------------------------------------------------------+
bool FiboArcPointChange(const long   chart_ID=0,      // ID графика
                        const string name="FiboArc", // имя объекта
                        const int    point_index=0,  // номер точки привязки
                        datetime     time=0,         // координата времени точки привя
                        double       price=0)        // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет "Дуги Фибоначчи"                                         |
//+------------------------------------------------------------------+
bool FiboArcDelete(const long   chart_ID=0,      // ID графика
                   const string name="FiboArc") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим объект
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Дуги Фибоначчи\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Дуг Фибоначчи" и для пустых   |
```

© 2000-2025, MetaQuotes Ltd.

```
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
void ChangeFiboArcEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- если время второй точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если цена второй точки не задана, то она будет иметь значение Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время первой точки не задано, то она лежит на 9 баров левее второй
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то сдвинем ее на 300 пунктов ниже второй
   if(!price1)
      price1=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки "Дуг Фибоначчи"
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования "Дуг Фибоначчи"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- создадим объект
   if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
      InpFullEllipse,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrd
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки
//--- счетчик цикла
   int v_steps=accuracy/5;
//--- перемещаем первую точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1<accuracy-1)
         p1+=1;
      //--- сдвигаем точку
      if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int h_steps=bars/5;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- перемещаем вторую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d2<bars-1)
         d2+=1;
      //--- сдвигаем точку
      if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим объект с графика
   FiboArcDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOCHANNEL

Канал Фибоначчи.



Примечание

Для "Канала Фибоначчи" можно указать режим продолжения его отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно).

Также можно указать количество линий-уровней, их значения и цвет.

Пример

Следующий скрипт создает и перемещает на графике "Канал Фибоначчи". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Канал Фибоначчи\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="FiboChannel";     // Имя канала
input int             InpDate1=20;               // Дата 1-ой точки в %
input int             InpPrice1=10;              // Цена 1-ой точки в %
input int             InpDate2=60;               // Дата 2-ой точки в %
input int             InpPrice2=30;              // Цена 2-ой точки в %
input int             InpDate3=20;               // Дата 3-ей точки в %
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpPrice3=25;              // Цена 3-ей точки в %
input color            InpColor=clrRed;           // Цвет канала
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Стиль линий канала
input int              InpWidth=2;                // Толщина линий канала
input bool             InpBack=false;             // Канал на заднем плане
input bool             InpSelection=true;         // Выделить для перемещений
input bool             InpRayLeft=false;          // Продолжение канала влево
input bool             InpRayRight=false;         // Продолжение канала вправо
input bool             InpHidden=true;            // Скрыт в списке объектов
input long             InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Канал Фибоначчи" по заданным координатам                |
//+------------------------------------------------------------------+
bool FiboChannelCreate(const long             chart_ID=0,          // ID графика
                       const string           name="FiboChannel", // имя канала
                       const int              sub_window=0,       // номер подокна
                       datetime               time1=0,            // время первой точки
                       double                 price1=0,           // цена первой точки
                       datetime               time2=0,            // время второй точки
                       double                 price2=0,           // цена второй точки
                       datetime               time3=0,            // время третьей точк
                       double                 price3=0,           // цена третьей точки
                       const color            clr=clrRed,         // цвет канала
                       const ENUM_LINE_STYLE  style=STYLE_SOLID,  // стиль линий канала
                       const int              width=1,            // толщина линий кана
                       const bool             back=false,         // на заднем плане
                       const bool             selection=true,     // выделить для перем
                       const bool             ray_left=false,     // продолжение канала
                       const bool             ray_right=false,    // продолжение канала
                       const bool             hidden=true,        // скрыт в списке об
                       const long             z_order=0)          // приоритет на нажат
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим канал по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Канал Фибоначчи\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет канала
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий канала
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий канала
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения канала для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения канала влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения канала вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Задает количество уровней и их параметры                         |
//+------------------------------------------------------------------+
bool FiboChannelLevelsSet(int             levels,             // количество линий уро
                          double          &values[],          // значения линий уровня
                          color           &colors[],          // цвет линий уровня
                          ENUM_LINE_STYLE &styles[],          // стиль линий уровня
                          int             &widths[],          // толщина линий уровня
                          const long      chart_ID=0,         // ID графика
                          const string    name="FiboChannel") // имя объекта
  {
//--- проверим размеры массивов
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": длина массива не соответствует количеству уровней, ошибка!
      return(false);
     }
//--- установим количество уровней
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- установим свойства уровней в цикле
   for(int i=0;i<levels;i++)
     {
      //--- значение уровня
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- цвет уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- стиль уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- толщина уровня
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- описание уровня
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Канала Фибоначчи"                     |
//+------------------------------------------------------------------+
bool FiboChannelPointChange(const long   chart_ID=0,          // ID графика
                            const string name="FiboChannel", // имя канала
                            const int    point_index=0,      // номер точки привязки
                            datetime     time=0,             // координата времени точ
                            double       price=0)            // координата цены точки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет канал                                                    |
//+------------------------------------------------------------------+
bool FiboChannelDelete(const long   chart_ID=0,          // ID графика
                       const string name="FiboChannel") // имя канала
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим канал
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Канал Фибоначчи\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Канала Фибоначчи" и для       |
//| пустых значений устанавливает значения по умолчанию              |
//+------------------------------------------------------------------+
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                  double &price2,datetime &time3,double &price3)
  {
//--- если время второй (правой) точки не задано, то она будет на текущем баре
   if(!time2)
      time2=TimeCurrent();
//--- если цена второй точки не задана, то она будет иметь значение Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время первой (левой) точки не задано, то она лежит на 9 баров левее второй
   if(!time1)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то сдвинем ее на 300 пунктов выше второй
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- если время третьей точки не задано, то оно совпадает с временем первой точки
   if(!time3)
      time3=time1;
//--- если цена третьей точки не задана, то она совпадает с ценой второй точки
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   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("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки канала
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования канала
   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;
//--- создадим "Канал Фибоначчи"
   if(!FiboChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки канала
//--- счетчик цикла
   int h_steps=bars/10;
//--- перемещаем первую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d1>1)
         d1-=1;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- сдвигаем точку
      if(!FiboChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int v_steps=accuracy/10;
//--- перемещаем вторую точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p2>1)
         p2-=1;
      //--- сдвигаем точку
      if(!FiboChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   v_steps=accuracy/15;
//--- перемещаем третью точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p3<accuracy-1)
         p3+=1;
      //--- сдвигаем точку
      if(!FiboChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
```

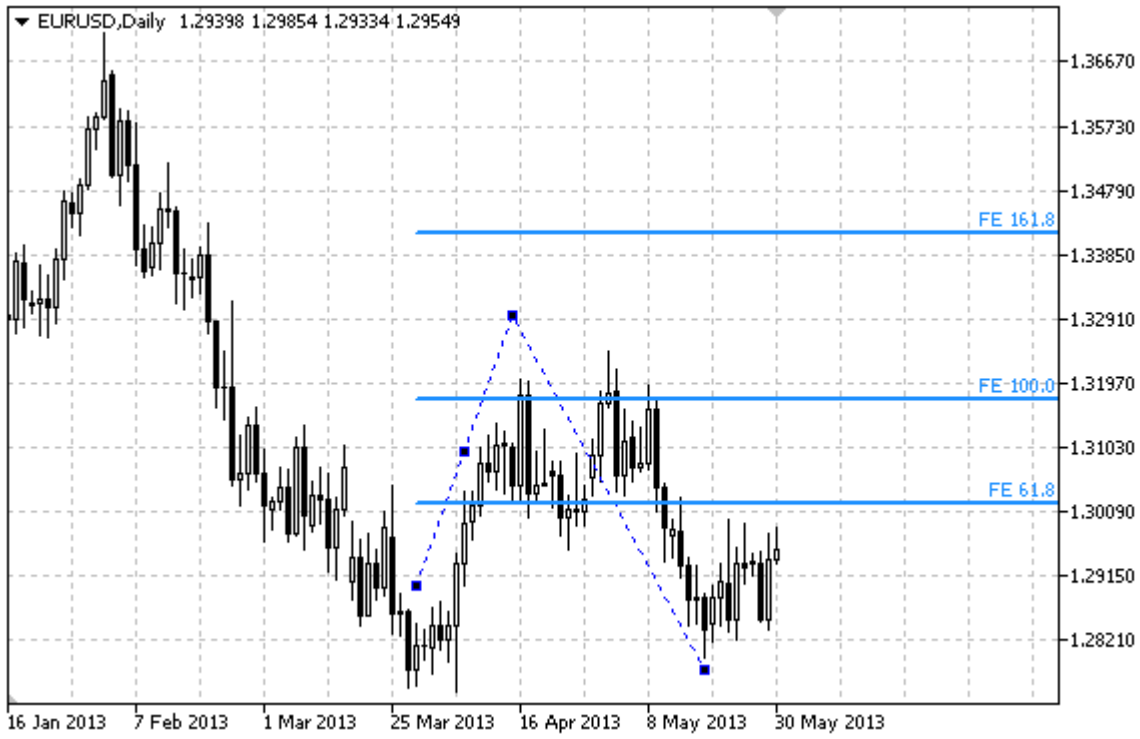
© 2000-2025, MetaQuotes Ltd.

```
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим канал с графика
   FiboChannelDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EXPANSION

Расширение Фибоначчи.



Примечание

Для "Расширения Фибоначчи" можно указать режим продолжения отображения вправо и/или влево (свойства OBJPROP_RAY_RIGHT и OBJPROP_RAY_LEFT соответственно).

Также можно указать количество линий-уровней, их значения и цвет.

Пример

Следующий скрипт создает и перемещает на графике "Расширение Фибоначчи". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Расширение Фибоначчи\"."
#property description "Координаты точек привязки задаются в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="FiboExpansion";   // Имя объекта
input int             InpDate1=10;               // Дата 1-ой точки в %
input int             InpPrice1=55;              // Цена 1-ой точки в %
input int             InpDate2=30;               // Дата 2-ой точки в %
input int             InpPrice2=10;              // Цена 2-ой точки в %
input int             InpDate3=80;               // Дата 3-ей точки в %
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpPrice3=75;              // Цена 3-ей точки в %
input color            InpColor=clrRed;           // Цвет объекта
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Стиль линий
input int              InpWidth=2;                // Толщина линий
input bool             InpBack=false;             // Объект на заднем плане
input bool             InpSelection=true;         // Выделить для перемещений
input bool             InpRayLeft=false;          // Продолжение объекта влево
input bool             InpRayRight=false;         // Продолжение объекта вправо
input bool             InpHidden=true;            // Скрыт в списке объектов
input long             InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Расширение Фибоначчи" по заданным координатам           |
//+------------------------------------------------------------------+
bool FiboExpansionCreate(const long             chart_ID=0,           // ID графика
                         const string           name="FiboExpansion", // имя канала
                         const int              sub_window=0,         // номер подокна
                         datetime               time1=0,              // время первой т
                         double                 price1=0,             // цена первой то
                         datetime               time2=0,              // время второй т
                         double                 price2=0,             // цена второй то
                         datetime               time3=0,              // время третьей
                         double                 price3=0,             // цена третьей т
                         const color            clr=clrRed,           // цвет объекта
                         const ENUM_LINE_STYLE  style=STYLE_SOLID,    // стиль линий
                         const int              width=1,              // толщина линий
                         const bool             back=false,           // на заднем план
                         const bool             selection=true,       // выделить для п
                         const bool             ray_left=false,       // продолжение об
                         const bool             ray_right=false,      // продолжение об
                         const bool             hidden=true,          // скрыт в списке
                         const long             z_order=0)            // приоритет на н
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Расширение Фибоначчи" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Расширение Фибоначчи\"! Код ошибки = ",GetLastErro
      return(false);
     }
//--- установим цвет объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения объекта для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- включим (true) или отключим (false) режим продолжения отображения объекта влево
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- включим (true) или отключим (false) режим продолжения отображения объекта вправо
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Задает количество уровней и их параметры                         |
//+------------------------------------------------------------------+
bool FiboExpansionLevelsSet(int             levels,                // количество линий
                            double          &values[],             // значения линий ур
                            color           &colors[],             // цвет линий уровня
                            ENUM_LINE_STYLE &styles[],             // стиль линий урове
                            int             &widths[],             // толщина линий уро
                            const long      chart_ID=0,            // ID графика
                            const string    name="FiboExpansion") // имя объекта
  {
//--- проверим размеры массивов
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": длина массива не соответствует количеству уровней, ошибка!
      return(false);
     }
//--- установим количество уровней
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- установим свойства уровней в цикле
   for(int i=0;i<levels;i++)
     {
      //--- значение уровня
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- цвет уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- стиль уровня
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- толщина уровня
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- описание уровня
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE "+DoubleToString(100*value
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Расширения Фибоначчи"                 |
//+------------------------------------------------------------------+
bool FiboExpansionPointChange(const long   chart_ID=0,           // ID графика
                              const string name="FiboExpansion", // имя объекта
                              const int    point_index=0,        // номер точки привяз
                              datetime     time=0,               // координата времени
                              double       price=0)              // координата цены то
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет "Расширение Фибоначчи"                                   |
//+------------------------------------------------------------------+
bool FiboExpansionDelete(const long   chart_ID=0,           // ID графика
                         const string name="FiboExpansion") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим объект
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Расширение Фибоначчи\"! Код ошибки = ",GetLastErro
      return(false);
     }
//--- успешное выполнение
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Расширения Фибоначчи" и для   |
//| пустых значений устанавливает значения по умолчанию              |
//+------------------------------------------------------------------+
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                    double &price2,datetime &time3,double &price3)
  {
//--- если время третьей (правой) точки не задано, то она будет на текущем баре
   if(!time3)
      time3=TimeCurrent();
//--- если цена третьей точки не задана, то она будет иметь значение Bid
   if(!price3)
      price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время первой (левой) точки не задано, то она лежит на 9 баров левее третьей
//--- массив для приема времени открытия 10 последних баров
   datetime temp[];
   ArrayResize(temp,10);
   if(!time1)
     {
      CopyTime(Symbol(),Period(),time3,10,temp);
      //--- установим первую точку на 9 баров левее второй
      time1=temp[0];
     }
//--- если цена первой точки не задана, то она совпадает с ценой третьей точки
   if(!price1)
      price1=price3;
//--- если время второй точки не задано, то она лежит на 7 баров левее третьей
   if(!time2)
      time2=temp[2];
//--- если цена второй точки не задана, сдвинем ее на 250 пунктов ниже первой
   if(!price2)
      price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   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("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
```

© 2000-2025, MetaQuotes Ltd.

```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки объекта
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования "Расширения Фибоначчи"
   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;
//--- создадим "Расширение Фибоначчи"
   if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки
//--- счетчик цикла
   int v_steps=accuracy/10;
//--- перемещаем первую точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1>1)
```

© 2000-2025, MetaQuotes Ltd.

```
         p1-=1;
      //--- сдвигаем точку
      if(!FiboExpansionPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   v_steps=accuracy/2;
//--- перемещаем третью точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p3>1)
         p3-=1;
      //--- сдвигаем точку
      if(!FiboExpansionPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   v_steps=accuracy*4/5;
//--- перемещаем вторую точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p2<accuracy-1)
         p2+=1;
      //--- сдвигаем точку
      if(!FiboExpansionPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(1000);
//--- удалим объект с графика
   FiboExpansionDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIOTWAVE5

Импульсная волна Эллиота.



Примечание

Для "Импульсной волны Эллиота" можно включить/отключить режим соединения точек линиями (свойство OBJPROP_DRAWLINES), а также установить уровень волновой разметки (из перечисления ENUM_ELLIOT_WAVE_DEGREE).

Пример

Следующий скрипт создает и перемещает на графике "Импульсную волну Эллиота". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Импульсная волна Эллиота\"."
#property description "Координаты точек привязки задаются в процентах от размеров"
#property description "окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string                InpName="ElliotWave5";   // Имя объекта
input int                   InpDate1=10;             // Дата 1-ой точки в %
input int                   InpPrice1=90;            // Цена 1-ой точки в %
input int                   InpDate2=20;             // Дата 2-ой точки в %
input int                   InpPrice2=40;            // Цена 2-ой точки в %
input int                   InpDate3=30;             // Дата 3-ей точки в %
```

© 2000-2025, MetaQuotes Ltd.

```
input int                      InpPrice3=60;            // Цена 3-ей точки в %
input int                      InpDate4=40;             // Дата 4-ой точки в %
input int                      InpPrice4=10;            // Цена 4-ой точки в %
input int                      InpDate5=60;             // Дата 5-ой точки в %
input int                      InpPrice5=40;            // Цена 5-ой точки в %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Уровень
input bool                     InpDrawLines=true;       // Отображение линий
input color                    InpColor=clrRed;         // Цвет линий
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Стиль линий
input int                      InpWidth=2;              // Толщина линий
input bool                     InpBack=false;           // Объект на заднем плане
input bool                     InpSelection=true;       // Выделить для перемещений
input bool                     InpHidden=true;          // Скрыт в списке объектов
input long                     InpZOrder=0;             // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Импульсную волну Эллиота" по заданным координатам       |
//+------------------------------------------------------------------+
bool ElliotWave5Create(const long                  chart_ID=0,               // ID гр
                       const string                name="ElliotWave5",       // имя к
                       const int                   sub_window=0,             // номер
                       datetime                    time1=0,                  // время
                       double                      price1=0,                 // цена
                       datetime                    time2=0,                  // время
                       double                      price2=0,                 // цена
                       datetime                    time3=0,                  // время
                       double                      price3=0,                 // цена
                       datetime                    time4=0,                  // время
                       double                      price4=0,                 // цена
                       datetime                    time5=0,                  // время
                       double                      price5=0,                 // цена
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // степе
                       const bool                  draw_lines=true,          // отобр
                       const color                 clr=clrRed,               // цвет
                       const ENUM_LINE_STYLE       style=STYLE_SOLID,        // стиль
                       const int                   width=1,                  // толщи
                       const bool                  back=false,               // на за
                       const bool                  selection=true,           // выдел
                       const bool                  hidden=true,              // скрыт
                       const long                  z_order=0)                // прио
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,ti
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Импульсную волну Эллиота" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,price2
      price3,time4,price4,time5,price5))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": не удалось создать \"Импульсную волну Эллиота\"! Код ошибки = ",GetLast
      return(false);
     }
//--- установим степень (размер волны)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- включим (true) или отключим (false) режим отображения линий
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- установим цвет объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения объекта для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Импульсной волны Эллиота"             |
//+------------------------------------------------------------------+
bool ElliotWave5PointChange(const long   chart_ID=0,          // ID графика
                            const string name="ElliotWave5", // имя объекта
                            const int    point_index=0,      // номер точки привязки
                            datetime     time=0,             // координата времени точ
                            double       price=0)            // координата цены точки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет "Импульсную волну Эллиота"                               |
//+------------------------------------------------------------------+
bool ElliotWave5Delete(const long   chart_ID=0,         // ID графика
                       const string name="ElliotWave5") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим объект
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Импульсную волну Эллиота\"! Код ошибки = ",GetLast
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Импульсной волны Эллиота"  и  |
//| для пустых значений устанавливает значения по умолчанию          |
//+------------------------------------------------------------------+
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3,
                                  datetime &time4,double &price4,
                                  datetime &time5,double &price5)
  {
//--- массив для приема времени открытия 10 последних баров
   datetime temp[];
   ArrayResize(temp,10);
//--- получим данные
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- получим величину одного пункта на текущем графике
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- если время первой точки не задано, то она будет на 9 баре слева от последнего ба
   if(!time1)
      time1=temp[0];
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время второй точки не задано, то она будет на 7 баре слева от последнего ба
   if(!time2)
      time2=temp[2];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если цена второй точки не задана, сдвинем ее на 300 пунктов ниже первой
   if(!price2)
      price2=price1-300*point;
//--- если время третьей точки не задано, то она будет на 5 баров слева от последнего
   if(!time3)
      time3=temp[4];
//--- если цена третьей точки не задана, сдвинем ее на 250 пунктов ниже первой
   if(!price3)
      price3=price1-250*point;
//--- если время четвертой точки не задано, то она будет на 3 бара слева от последнего
   if(!time4)
      time4=temp[6];
//--- если цена четвертой точки не задана, сдвинем ее на 550 пунктов ниже первой
   if(!price4)
      price4=price1-550*point;
//--- если время пятой точки не задано, то она будет на последнем баре
   if(!time5)
      time5=temp[9];
//--- если цена пятой точки не задана, сдвинем ее на 450 пунктов ниже первой
   if(!price5)
      price5=price1-450*point;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   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("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки объекта
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования "Импульсной волны Эллиота"
   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;
//--- создадим "Импульсную волну Эллиота"
   if(!ElliotWave5Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      date[d4],price[p4],date[d5],price[p5],InpDegree,InpDrawLines,InpColor,InpStyle,I
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки
//--- счетчик цикла
   int v_steps=accuracy/5;
//--- перемещаем пятую точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p5<accuracy-1)
         p5+=1;
      //--- сдвигаем точку
      if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
       //--- перерисуем график
       ChartRedraw();
      }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   v_steps=accuracy/5;
//--- перемещаем вторую и третью точки привязки
   for(int i=0;i<v_steps;i++)
     {
       //--- возьмем следующие значения
       if(p2<accuracy-1)
          p2+=1;
       if(p3>1)
          p3-=1;
       //--- сдвигаем точки
       if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
          return;
       if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
          return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
          return;
       //--- перерисуем график
       ChartRedraw();
      }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   v_steps=accuracy*4/5;
//--- перемещаем первую и четвертую точки привязки
   for(int i=0;i<v_steps;i++)
     {
       //--- возьмем следующие значения
       if(p1>1)
          p1-=1;
       if(p4<accuracy-1)
          p4+=1;
       //--- сдвигаем точки
       if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
          return;
       if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
          return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
          return;
       //--- перерисуем график
       ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим объект с графика
   ElliotWave5Delete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIOTWAVE3

Корректирующая волна Эллиота.



Примечание

Для "Корректирующей волны Эллиота" можно включить/отключить режим соединения точек линиями (свойство OBJPROP_DRAWLINES), а также установить уровень волновой разметки (из перечисления ENUM_ELLIOT_WAVE_DEGREE).

Пример

Следующий скрипт создает и перемещает на графике "Корректирующую волну Эллиота". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Корректирующая волна Эллиота
#property description "Координаты точек привязки задаются в процентах от размеров окна
#property description "графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string                InpName="ElliotWave3";   // Имя объекта
input int                   InpDate1=10;             // Дата 1-ой точки в %
input int                   InpPrice1=90;            // Цена 1-ой точки в %
input int                   InpDate2=30;             // Дата 2-ой точки в %
input int                   InpPrice2=10;            // Цена 2-ой точки в %
input int                   InpDate3=50;             // Дата 3-ей точки в %
```

© 2000-2025, MetaQuotes Ltd.

```
input int                      InpPrice3=40;            // Цена 3-ей точки в %
input ENUM_ELLIOT_WAVE_DEGREE  InpDegree=ELLIOTT_MINOR; // Уровень
input bool                     InpDrawLines=true;       // Отображение линий
input color                    InpColor=clrRed;         // Цвет линий
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Стиль линий
input int                      InpWidth=2;              // Толщина линий
input bool                     InpBack=false;           // Объект на заднем плане
input bool                     InpSelection=true;       // Выделить для перемещений
input bool                     InpHidden=true;          // Скрыт в списке объектов
input long                     InpZOrder=0;             // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает "Корректирующую волну Эллиота" по заданным координатам   |
//+------------------------------------------------------------------+
bool ElliotWave3Create(const long                  chart_ID=0,               // ID гр
                       const string                name="ElliotWave3",       // имя в
                       const int                   sub_window=0,             // номер
                       datetime                    time1=0,                  // время
                       double                      price1=0,                 // цена
                       datetime                    time2=0,                  // время
                       double                      price2=0,                 // цена
                       datetime                    time3=0,                  // время
                       double                      price3=0,                 // цена
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // степе
                       const bool                  draw_lines=true,          // отобр
                       const color                 clr=clrRed,               // цвет
                       const ENUM_LINE_STYLE       style=STYLE_SOLID,        // стиль
                       const int                   width=1,                  // толщи
                       const bool                  back=false,               // на за
                       const bool                  selection=true,           // выдел
                       const bool                  hidden=true,              // скрыт
                       const long                  z_order=0)                // прио
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим "Корректирующую волну Эллиота" по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": не удалось создать \"Корректирующую волну Эллиота\"! Код ошибки = ",Get
      return(false);
     }
//--- установим степень (размер волны)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- включим (true) или отключим (false) режим отображения линий
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- установим цвет объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим стиль линий
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения объекта для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки "Корректирующей волны Эллиота"         |
//+------------------------------------------------------------------+
bool ElliotWave3PointChange(const long   chart_ID=0,          // ID графика
                            const string name="ElliotWave3", // имя объекта
                            const int    point_index=0,      // номер точки привязки
                            datetime     time=0,             // координата времени точ
                            double       price=0)            // координата цены точки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет "Корректирующую волну Эллиота"                           |
//+------------------------------------------------------------------+
bool ElliotWave3Delete(const long   chart_ID=0,          // ID графика
```

© 2000-2025, MetaQuotes Ltd.

```
                        const string name="ElliotWave3") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим объект
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить \"Корректирующую волну Эллиота\"! Код ошибки = ",Get
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки "Корректирующей волны Эллиота" |
//| и для пустых значений устанавливает значения по умолчанию        |
//+------------------------------------------------------------------+
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3)
  {
//--- массив для приема времени открытия 10 последних баров
   datetime temp[];
   ArrayResize(temp,10);
//--- получим данные
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- получим величину одного пункта на текущем графике
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- если время первой точки не задано, то она будет на 9 баре слева от последнего ба
   if(!time1)
      time1=temp[0];
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время второй точки не задано, то она будет на 5 баре слева от последнего ба
   if(!time2)
      time2=temp[4];
//--- если цена второй точки не задана, сдвинем ее на 300 пунктов ниже первой
   if(!price2)
      price2=price1-300*point;
//--- если время третьей точки не задано, то она будет на 1 баре слева от последнего б
   if(!time3)
      time3=temp[8];
//--- если цена третьей точки не задана, сдвинем ее на 200 пунктов ниже первой
   if(!price3)
      price3=price1-200*point;
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   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("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки объекта
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования "Корректирующей волны Эллиота"
   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;
//--- создадим "Корректирующую волну Эллиота"
   if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p
      InpDegree,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidde
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки
//--- счетчик цикла
   int v_steps=accuracy/5;
//--- перемещаем третью точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p3<accuracy-1)
         p3+=1;
      //--- сдвигаем точку
      if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   v_steps=accuracy*4/5;
//--- перемещаем первую и вторую точки привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующие значения
      if(p1>1)
         p1-=1;
      if(p2<accuracy-1)
         p2+=1;
      //--- сдвигаем точки
      if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим объект с графика
```

© 2000-2025, MetaQuotes Ltd.

```
   ElliotWave3Delete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_RECTANGLE

Прямоугольник.



Примечание

Для прямоугольника можно установить режим заливки при помощи свойства OBJPROP_FILL.

Пример

Следующий скрипт создает и перемещает на графике прямоугольник. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит прямоугольник на графике."
#property description "Координаты точек привязки задаются в"
#property description "процентах от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="Rectangle"; // Имя прямоугольника
input int             InpDate1=40;         // Дата 1-ой точки в %
input int             InpPrice1=40;        // Цена 1-ой точки в %
input int             InpDate2=60;         // Дата 2-ой точки в %
input int             InpPrice2=60;        // Цена 2-ой точки в %
input color           InpColor=clrRed;     // Цвет прямоугольника
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Стиль линий прямоугольника
input int             InpWidth=2;          // Толщина линий прямоугольника
```

© 2000-2025, MetaQuotes Ltd.

```
input bool             InpFill=true;         // Заливка прямоугольника цветом
input bool             InpBack=false;        // Прямоугольника на заднем плане
input bool             InpSelection=true;    // Выделить для перемещений
input bool             InpHidden=true;       // Скрыт в списке объектов
input long             InpZOrder=0;          // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает прямоугольник по заданным координатам                    |
//+------------------------------------------------------------------+
bool RectangleCreate(const long            chart_ID=0,        // ID графика
                     const string          name="Rectangle",  // имя прямоугольника
                     const int             sub_window=0,      // номер подокна
                     datetime              time1=0,           // время первой точки
                     double                price1=0,          // цена первой точки
                     datetime              time2=0,           // время второй точки
                     double                price2=0,          // цена второй точки
                     const color           clr=clrRed,        // цвет прямоугольника
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линий прямоугол
                     const int             width=1,           // толщина линий прямоуг
                     const bool            fill=false,        // заливка прямоугольник
                     const bool            back=false,        // на заднем плане
                     const bool            selection=true,    // выделить для перемеще
                     const bool            hidden=true,       // скрыт в списке объект
                     const long            z_order=0)         // приоритет на нажатие
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим прямоугольник по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": не удалось создать прямоугольник! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет прямоугольника
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий прямоугольника
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий прямоугольника
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- включим (true) или отключим (false) режим заливки прямоугольника
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения прямоугольника для перемеще
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки прямоугольника                         |
//+------------------------------------------------------------------+
bool RectanglePointChange(const long   chart_ID=0,        // ID графика
                          const string name="Rectangle", // имя прямоугольника
                          const int    point_index=0,    // номер точки привязки
                          datetime     time=0,           // координата времени точки
                          double       price=0)          // координата цены точки прив
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет прямоугольник                                            |
//+------------------------------------------------------------------+
bool RectangleDelete(const long   chart_ID=0,        // ID графика
                     const string name="Rectangle") // имя прямоугольника
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим прямоугольник
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить прямоугольник! Код ошибки = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки прямоугольника и для пустых    |
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- если время первой точки не задано, то она будет на текущем баре
   if(!time1)
      time1=TimeCurrent();
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время второй точки не задано, то она лежит на 9 баров левее второй
   if(!time2)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- установим вторую точку на 9 баров левее первой
      time2=temp[0];
     }
//--- если цена второй точки не задана, сдвинем ее на 300 пунктов ниже первой
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки прямоугольника
   datetime date[];
```

© 2000-2025, MetaQuotes Ltd.

```
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования прямоугольника
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- создадим прямоугольник
   if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки прямоугольника
//--- счетчик цикла
   int h_steps=bars/2;
//--- перемещаем точки привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующие значения
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- сдвигаем точки
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int v_steps=accuracy/2;
//--- перемещаем точки привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующие значения
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- сдвигаем точки
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим прямоугольник с графика
   RectangleDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TRIANGLE

Треугольник.



Примечание

Для треугольника можно установить режим заливки при помощи свойства OBJPROP_FILL.

Пример

Следующий скрипт создает и перемещает на графике треугольник. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит треугольник на графике."
#property description "Координаты точек привязки задаются в"
#property description "процентах от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="Triangle";         // Имя треугольника
input int              InpDate1=25;                // Дата 1-ой точки в %
input int              InpPrice1=50;               // Цена 1-ой точки в %
input int              InpDate2=70;                // Дата 2-ой точки в %
input int              InpPrice2=70;               // Цена 2-ой точки в %
input int              InpDate3=65;                // Дата 3-ей точки в %
input int              InpPrice3=20;               // Цена 3-ей точки в %
input color            InpColor=clrRed;            // Цвет треугольника
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Стиль линий треугольника
input int             InpWidth=2;                // Толщина линий треугольника
input bool            InpFill=false;             // Заливка треугольника цветом
input bool            InpBack=false;             // Треугольника на заднем плане
input bool            InpSelection=true;         // Выделить для перемещений
input bool            InpHidden=true;            // Скрыт в списке объектов
input long            InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает треугольник по заданным координатам                      |
//+------------------------------------------------------------------+
bool TriangleCreate(const long            chart_ID=0,        // ID графика
                    const string          name="Triangle",   // имя треугольника
                    const int             sub_window=0,      // номер подокна
                    datetime              time1=0,           // время первой точки
                    double                price1=0,          // цена первой точки
                    datetime              time2=0,           // время второй точки
                    double                price2=0,          // цена второй точки
                    datetime              time3=0,           // время третьей точки
                    double                price3=0,          // цена третьей точки
                    const color           clr=clrRed,        // цвет треугольника
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линий треугольни
                    const int             width=1,           // толщина линий треуголь
                    const bool            fill=false,        // заливка треугольника ц
                    const bool            back=false,        // на заднем плане
                    const bool            selection=true,    // выделить для перемещен
                    const bool            hidden=true,       // скрыт в списке объекто
                    const long            z_order=0)         // приоритет на нажатие м
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим треугольник по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,ti
     {
      Print(__FUNCTION__,
            ": не удалось создать треугольник! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет треугольника
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий треугольника
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий треугольника
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- включим (true) или отключим (false) режим заливки треугольника
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- включим (true) или отключим (false) режим выделения треугольник для перемещений
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки треугольника                           |
//+------------------------------------------------------------------+
bool TrianglePointChange(const long   chart_ID=0,       // ID графика
                         const string name="Triangle", // имя треугольника
                         const int    point_index=0,   // номер точки привязки
                         datetime     time=0,          // координата времени точки при
                         double       price=0)         // координата цены точки привяз
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет треугольник                                              |
//+------------------------------------------------------------------+
bool TriangleDelete(const long   chart_ID=0,       // ID графика
                    const string name="Triangle") // имя треугольника
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим треугольник
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": не удалось удалить треугольник! Код ошибки = ",GetLastError());
       return(false);
      }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки треугольника и для пустых      |
//| значений устанавливает значения по умолчанию                     |
//+------------------------------------------------------------------+
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                               datetime &time2,double &price2,
                               datetime &time3,double &price3)
  {
//--- если время первой точки не задано, то она будет на текущем баре
   if(!time1)
      time1=TimeCurrent();
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время второй точки не задано, то она лежит на 9 баров левее второй
   if(!time2)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- установим вторую точку на 9 баров левее первой
      time2=temp[0];
     }
//--- если цена второй точки не задана, сдвинем ее на 300 пунктов ниже первой
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- если время третьей точки не задано, то оно будет совпадать с датой второй точки
   if(!time3)
      time3=time2;
//--- если цена первой точки не задана, то она будет совпадать с ценой первой точки
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
```

© 2000-2025, MetaQuotes Ltd.

```
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки треугольника
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования треугольника
   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;
//--- создадим треугольник
   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;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки треугольника
//--- счетчик цикла
```

© 2000-2025, MetaQuotes Ltd.

```
   int v_steps=accuracy*3/10;
//--- перемещаем первую точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p1>1)
         p1-=1;
      //--- сдвигаем точку
      if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int h_steps=bars*9/20-1;
//--- перемещаем вторую точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d2>1)
         d2-=1;
      //--- сдвигаем точку
      if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   v_steps=accuracy/4;
//--- перемещаем третью точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p3<accuracy-1)
         p3+=1;
      //--- сдвигаем точку
      if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим треугольник с графика
   TriangleDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIPSE

Эллипс.



Примечание

Для эллипса можно установить режим заливки при помощи свойства OBJPROP_FILL.

Пример

Следующий скрипт создает и перемещает на графике эллипс. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит эллипс на графике."
#property description "Координаты точек привязки задаются"
#property description "в процентах от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpName="Ellipse";         // Имя эллипса
input int             InpDate1=30;               // Дата 1-ой точки в %
input int             InpPrice1=20;              // Цена 1-ой точки в %
input int             InpDate2=70;               // Дата 2-ой точки в %
input int             InpPrice2=80;              // Цена 2-ой точки в %
input int             InpDate3=50;               // Дата 3-ей точки в %
input int             InpPrice3=60;              // Цена 3-ей точки в %
input color           InpColor=clrRed;           // Цвет эллипса
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Стиль линий эллипса
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpWidth=2;                // Толщина линий эллипса
input bool             InpFill=false;             // Заливка эллипса цветом
input bool             InpBack=false;             // Эллипс на заднем плане
input bool             InpSelection=true;         // Выделить для перемещений
input bool             InpHidden=true;            // Скрыт в списке объектов
input long             InpZOrder=0;               // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает эллипс по заданным координатам                           |
//+------------------------------------------------------------------+
bool EllipseCreate(const long            chart_ID=0,        // ID графика
                   const string          name="Ellipse",    // имя эллипса
                   const int             sub_window=0,      // номер подокна
                   datetime              time1=0,           // время первой точки
                   double                price1=0,          // цена первой точки
                   datetime              time2=0,           // время второй точки
                   double                price2=0,          // цена второй точки
                   datetime              time3=0,           // время третьей точки
                   double                price3=0,          // цена третьей точки
                   const color           clr=clrRed,        // цвет эллипса
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линий эллипса
                   const int             width=1,           // толщина линий эллипса
                   const bool            fill=false,        // заливка эллипса цветом
                   const bool            back=false,        // на заднем плане
                   const bool            selection=true,    // выделить для перемещени
                   const bool            hidden=true,       // скрыт в списке объектов
                   const long            z_order=0)         // приоритет на нажатие мы
  {
//--- установим координаты точек привязки, если они не заданы
   ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим эллипс по заданным координатам
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,tim
     {
      Print(__FUNCTION__,
            ": не удалось создать эллипс! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет эллипса
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линий эллипса
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину линий эллипса
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- включим (true) или отключим (false) режим заливки эллипса
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим выделения эллипса для перемещений
```

© 2000-2025, MetaQuotes Ltd.

```
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки эллипса                                |
//+------------------------------------------------------------------+
bool EllipsePointChange(const long   chart_ID=0,     // ID графика
                        const string name="Ellipse", // имя эллипса
                        const int    point_index=0,  // номер точки привязки
                        datetime     time=0,         // координата времени точки привя
                        double       price=0)        // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет эллипс                                                   |
//+------------------------------------------------------------------+
bool EllipseDelete(const long   chart_ID=0,     // ID графика
                   const string name="Ellipse") // имя эллипса
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим эллипс
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": не удалось удалить эллипс! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точек привязки эллипса и для пустых значений  |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
  {
//--- если время первой точки не задано, то она будет на текущем баре
   if(!time1)
      time1=TimeCurrent();
//--- если цена первой точки не задана, то она будет иметь значение Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- если время второй точки не задано, то она лежит на 9 баров левее второй
   if(!time2)
     {
      //--- массив для приема времени открытия 10 последних баров
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- установим вторую точку на 9 баров левее первой
      time2=temp[0];
     }
//--- если цена второй точки не задана, сдвинем ее на 300 пунктов ниже первой
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- если время третьей точки не задано, то оно будет совпадать с датой второй точки
   if(!time3)
      time3=time2;
//--- если цена третьей точки не задана, то она будет совпадать с ценой первой точки
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   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)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("Ошибка! Некорректные значения входных параметров!");
       return;
      }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точек привязки эллипса
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования эллипса
   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;
//--- создадим эллипс
   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;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точки привязки эллипса
//--- счетчик цикла
   int v_steps=accuracy/5;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- перемещаем первую и вторую точки привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующие значения
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- сдвигаем точки
      if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int h_steps=bars/5;
//--- перемещаем третью точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d3>1)
         d3-=1;
      //--- сдвигаем точку
      if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим эллипс с графика
   EllipseDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_THUMB_UP

Знак "Хорошо".



Примечание

Положение точки привязки относительно знака можно выбрать из перечисления ENUM_ARROW_ANCHOR.

Знаки большого размера (больше 5) можно создать только установив соответствующее значение свойства OBJPROP_WIDTH при написании кода в MetaEditor.

Пример

Следующий скрипт создает и перемещает на графике знак "Хорошо". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт рисует знак \"Хорошо\" (большой палец вверх)."
#property description "Координата точки привязки задается в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string             InpName="ThumbUp";     // Имя знака
input int                InpDate=75;            // Дата точки привязки в %
input int                InpPrice=25;           // Цена точки привязки в %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;  // Способ привязки
input color              InpColor=clrRed;       // Цвет знака
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Стиль окаймляющей линии
input int               InpWidth=5;           // Размер знака
input bool              InpBack=false;        // Знак на заднем плане
input bool              InpSelection=true;    // Выделить для перемещений
input bool              InpHidden=true;       // Скрыт в списке объектов
input long              InpZOrder=0;          // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает знак "Хорошо"                                            |
//+------------------------------------------------------------------+
bool ArrowThumbUpCreate(const long              chart_ID=0,           // ID графика
                        const string            name="ThumbUp",       // имя знака
                        const int               sub_window=0,         // номер подокна
                        datetime                time=0,               // время точки
                        double                  price=0,              // цена точки пр
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // способ привяз
                        const color             clr=clrRed,           // цвет знака
                        const ENUM_LINE_STYLE   style=STYLE_SOLID,    // стиль окаймля
                        const int               width=3,              // размер знака
                        const bool              back=false,           // на заднем пла
                        const bool              selection=true,       // выделить для
                        const bool              hidden=true,          // скрыт в списк
                        const long              z_order=0)            // приоритет на
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим знак
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать знак \"Хорошо\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет знака
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль окаймляющей линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер знака
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения знака мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool ArrowThumbUpMove(const long   chart_ID=0,      // ID графика
                      const string name="ThumbUp", // имя объекта
                      datetime     time=0,         // координата времени точки привязк
                      double       price=0)        // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Меняет способ привязки знака "Хорошо"                            |
//+------------------------------------------------------------------+
bool ArrowThumbUpAnchorChange(const long              chart_ID=0,        // ID графика
                              const string            name="ThumbUp",    // имя объект
                              const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // способ при
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим способ привязки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": не удалось изменить способ привязки! Код ошибки = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет знак "Хорошо"                                            |
//+------------------------------------------------------------------+
bool ArrowThumbUpDelete(const long   chart_ID=0,     // ID графика
                        const string name="ThumbUp") // имя знака
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим знак
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить знак \"Хорошо\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точки привязки знака
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования знака
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- создадим знак "Хорошо" на графике
   if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки и менять ее положение относительно знака
//--- счетчик цикла
   int h_steps=bars/4;
//--- перемещаем точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d>1)
         d-=1;
      //--- сдвигаем точку
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int v_steps=accuracy/4;
//--- перемещаем точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p<accuracy-1)
         p+=1;
      //--- сдвигаем точку
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- меняем положение точки привязки относительно знака
   ArrowThumbUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- перерисуем график
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим знак с графика
   ArrowThumbUpDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_THUMB_DOWN

Знак "Плохо".



Примечание

Положение точки привязки относительно знака можно выбрать из перечисления ENUM_ARROW_ANCHOR.

Знаки большого размера (больше 5) можно создать только установив соответствующее значение свойства OBJPROP_WIDTH при написании кода в MetaEditor.

Пример

Следующий скрипт создает и перемещает на графике знак "Плохо". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт рисует знак \"Плохо\" (большой палец вниз)."
#property description "Координата точки привязки задается в процентах от"
#property description "размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string            InpName="ThumbDown";      // Имя знака
input int               InpDate=25;               // Дата точки привязки в %
input int               InpPrice=75;              // Цена точки привязки в %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Способ привязки
```

© 2000-2025, MetaQuotes Ltd.

```
input color             InpColor=clrRed;         // Цвет знака
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Стиль окаймляющей линии
input int               InpWidth=5;              // Размер знака
input bool              InpBack=false;           // Знак на заднем плане
input bool              InpSelection=true;       // Выделить для перемещений
input bool              InpHidden=true;          // Скрыт в списке объектов
input long              InpZOrder=0;             // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает знак "Плохо"                                             |
//+------------------------------------------------------------------+
bool ArrowThumbDownCreate(const long              chart_ID=0,           // ID графика
                          const string            name="ThumbDown",     // имя знака
                          const int               sub_window=0,         // номер подок
                          datetime                time=0,               // время точки
                          double                  price=0,              // цена точки
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // способ прив
                          const color             clr=clrRed,           // цвет знака
                          const ENUM_LINE_STYLE   style=STYLE_SOLID,    // стиль окайм
                          const int               width=3,              // размер знак
                          const bool              back=false,           // на заднем п
                          const bool              selection=true,       // выделить дл
                          const bool              hidden=true,          // скрыт в спи
                          const long              z_order=0)            // приоритет н
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим знак
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать знак \"Плохо\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет знака
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль окаймляющей линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер знака
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения знака мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool ArrowThumbDownMove(const long   chart_ID=0,        // ID графика
                        const string name="ThumbDown", // имя объекта
                        datetime     time=0,           // координата времени точки при
                        double       price=0)          // координата цены точки привяз
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Меняет способ привязки знака "Плохо"                             |
//+------------------------------------------------------------------+
bool ArrowThumbDownAnchorChange(const long              chart_ID=0,         // ID графи
                                const string            name="ThumbDown",   // имя объе
                                const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // способ и
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим способ привязки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": не удалось изменить способ привязки! Код ошибки = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет знак "Плохо"                                             |
//+------------------------------------------------------------------+
bool ArrowThumbDownDelete(const long   chart_ID=0,       // ID графика
                          const string name="ThumbDown") // имя знака
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим знак
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить знак \"Плохо\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
```

© 2000-2025, MetaQuotes Ltd.

```
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точки привязки знака
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования знака
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- создадим знак "Плохо" на графике
   if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки и менять ее положение относительно знака
//--- счетчик цикла
   int h_steps=bars/4;
//--- перемещаем точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d<bars-1)
         d+=1;
      //--- сдвигаем точку
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
       //--- перерисуем график
       ChartRedraw();
       // задержка в 0.05 секунды
       Sleep(50);
      }
//--- задержка в 1 секунду
   Sleep(1000);
//--- счетчик цикла
   int v_steps=accuracy/4;
//--- перемещаем точку привязки
   for(int i=0;i<v_steps;i++)
     {
       //--- возьмем следующее значение
       if(p>1)
         p-=1;
       //--- сдвигаем точку
       if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
         return;
       //--- перерисуем график
       ChartRedraw();
      }
//--- меняем положение точки привязки относительно знака
   ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- перерисуем график
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим знак с графика
   ArrowThumbDownDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_UP

Знак "Стрелка вверх".



Примечание

Положение точки привязки относительно знака можно выбрать из перечисления ENUM_ARROW_ANCHOR.

Знаки большого размера (больше 5) можно создать только установив соответствующее значение свойства OBJPROP_WIDTH при написании кода в MetaEditor.

Пример

Следующий скрипт создает и перемещает на графике знак "Стрелка вверх". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт рисует знак \"Стрелка вверх\"."
#property description "Координата точки привязки задается в"
#property description "процентах от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string             InpName="ArrowUp";     // Имя знака
input int                InpDate=25;            // Дата точки привязки в %
input int                InpPrice=25;           // Цена точки привязки в %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;  // Способ привязки
input color              InpColor=clrRed;       // Цвет знака
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Стиль окаймляющей линии
input int               InpWidth=5;           // Размер знака
input bool              InpBack=false;        // Знак на заднем плане
input bool              InpSelection=false;   // Выделить для перемещений
input bool              InpHidden=true;       // Скрыт в списке объектов
input long              InpZOrder=0;          // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает знак "Стрелка вверх"                                     |
//+------------------------------------------------------------------+
bool ArrowUpCreate(const long              chart_ID=0,           // ID графика
                   const string            name="ArrowUp",       // имя знака
                   const int               sub_window=0,         // номер подокна
                   datetime                time=0,               // время точки привяз
                   double                  price=0,              // цена точки привязк
                   const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // способ привязки
                   const color             clr=clrRed,           // цвет знака
                   const ENUM_LINE_STYLE   style=STYLE_SOLID,    // стиль окаймляющей
                   const int               width=3,              // размер знака
                   const bool              back=false,           // на заднем плане
                   const bool              selection=true,       // выделить для перем
                   const bool              hidden=true,          // скрыт в списке об
                   const long              z_order=0)            // приоритет на нажат
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим знак
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать знак \"Стрелка вверх\"! Код ошибки = ",GetLastError
      return(false);
     }
//--- установим способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет знака
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль окаймляющей линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер знака
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения знака мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool ArrowUpMove(const long   chart_ID=0,      // ID графика
                 const string name="ArrowUp", // имя объекта
                 datetime     time=0,         // координата времени точки привязки
                 double       price=0)        // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Меняет способ привязки знака "Стрелка вверх"                     |
//+------------------------------------------------------------------+
bool ArrowUpAnchorChange(const long              chart_ID=0,        // ID графика
                         const string            name="ArrowUp",    // имя объекта
                         const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // способ привязки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим положение точки привязки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": не удалось изменить способ привязки! Код ошибки = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет знак "Стрелка вверх"                                     |
//+------------------------------------------------------------------+
bool ArrowUpDelete(const long   chart_ID=0,     // ID графика
                   const string name="ArrowUp") // имя знака
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим знак
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить знак \"Стрелка вверх\"! Код ошибки = ",GetLastError
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точки привязки знака
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования знака
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- создадим знак "Стрелка вверх" на графике
   if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки и менять ее положение относительно знака
//--- счетчик цикла
   int v_steps=accuracy/2;
//--- перемещаем точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p<accuracy-1)
         p+=1;
      //--- сдвигаем точку
      if(!ArrowUpMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- меняем положение точки привязки относительно знака
   ArrowUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- перерисуем график
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим знак с графика
   ArrowUpDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_DOWN

Знак "Стрелка вниз".



Примечание

Положение точки привязки относительно знака можно выбрать из перечисления ENUM_ARROW_ANCHOR.

Знаки большого размера (больше 5) можно создать только установив соответствующее значение свойства OBJPROP_WIDTH при написании кода в MetaEditor.

Пример

Следующий скрипт создает и перемещает на графике знак "Стрелка вниз". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт рисует знак \"Стрелка вниз\"."
#property description "Координата точки привязки задается в"
#property description "процентах от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string            InpName="ArrowDown";      // Имя знака
input int               InpDate=75;               // Дата точки привязки в %
input int               InpPrice=75;              // Цена точки привязки в %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Способ привязки
input color             InpColor=clrRed;          // Цвет знака
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;       // Стиль окаймляющей линии
```

© 2000-2025, MetaQuotes Ltd.

```
input int                InpWidth=5;               // Размер знака
input bool               InpBack=false;            // Знак на заднем плане
input bool               InpSelection=false;       // Выделить для перемещений
input bool               InpHidden=true;           // Скрыт в списке объектов
input long               InpZOrder=0;              // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает знак "Стрелка вниз"                                      |
//+------------------------------------------------------------------+
bool ArrowDownCreate(const long              chart_ID=0,           // ID графика
                     const string            name="ArrowDown",     // имя знака
                     const int               sub_window=0,         // номер подокна
                     datetime                time=0,               // время точки прив
                     double                  price=0,              // цена точки привя
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // способ привязки
                     const color             clr=clrRed,           // цвет знака
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // стиль окаймляюще
                     const int               width=3,              // размер знака
                     const bool              back=false,           // на заднем плане
                     const bool              selection=true,       // выделить для пер
                     const bool              hidden=true,          // скрыт в списке о
                     const long              z_order=0)            // приоритет на нах
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим знак
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать знак \"Стрелка вниз\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет знака
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль окаймляющей линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер знака
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения знака мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool ArrowDownMove(const long   chart_ID=0,        // ID графика
                   const string name="ArrowDown", // имя объекта
                   datetime     time=0,           // координата времени точки привязки
                   double       price=0)          // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Меняет способ привязки знака "Стрелка вниз"                      |
//+------------------------------------------------------------------+
bool ArrowDownAnchorChange(const long              chart_ID=0,         // ID графика
                           const string            name="ArrowDown",   // имя объекта
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // способ привяз
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим положение точки привязки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": не удалось изменить способ привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет знак "Стрелка вниз"                                      |
//+------------------------------------------------------------------+
bool ArrowDownDelete(const long   chart_ID=0,        // ID графика
                     const string name="ArrowDown") // имя знака
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим знак
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить знак \"Стрелка вниз\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
```

© 2000-2025, MetaQuotes Ltd.

```
//--- для установки и изменения координат точки привязки знака
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования знака
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- создадим знак "Стрелка вниз" на графике
   if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки и менять ее положение относительно знака
//--- счетчик цикла
   int v_steps=accuracy/2;
//--- перемещаем точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p>1)
         p-=1;
      //--- сдвигаем точку
      if(!ArrowDownMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
```

© 2000-2025, MetaQuotes Ltd.

```
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- меняем положение точки привязки относительно знака
   ArrowDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- перерисуем график
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим знак с графика
   ArrowDownDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_STOP

Знак "Стоп".



Примечание

Положение точки привязки относительно знака можно выбрать из перечисления ENUM_ARROW_ANCHOR.

Знаки большого размера (больше 5) можно создать только установив соответствующее значение свойства OBJPROP_WIDTH при написании кода в MetaEditor.

Пример

Следующий скрипт создает и перемещает на графике знак "Стоп". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт рисует знак \"Стоп\"."
#property description "Координата точки привязки задается в"
#property description "процентах от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string             InpName="ArrowStop";      // Имя знака
input int                InpDate=10;               // Дата точки привязки в %
input int                InpPrice=50;              // Цена точки привязки в %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // Способ привязки
input color              InpColor=clrRed;          // Цвет знака
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Стиль окаймляющей линии
input int               InpWidth=5;              // Размер знака
input bool              InpBack=false;           // Знак на заднем плане
input bool              InpSelection=false;      // Выделить для перемещений
input bool              InpHidden=true;          // Скрыт в списке объектов
input long              InpZOrder=0;             // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает знак "Стоп"                                              |
//+------------------------------------------------------------------+
bool ArrowStopCreate(const long              chart_ID=0,           // ID графика
                     const string            name="ArrowStop",     // имя знака
                     const int               sub_window=0,         // номер подокна
                     datetime                time=0,               // время точки прив
                     double                  price=0,              // цена точки привя
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // способ привязки
                     const color             clr=clrRed,           // цвет знака
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // стиль окаймляюще
                     const int               width=3,              // размер знака
                     const bool              back=false,           // на заднем плане
                     const bool              selection=true,       // выделить для пер
                     const bool              hidden=true,          // скрыт в списке с
                     const long              z_order=0)            // приоритет на нах
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим знак
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать знак \"Стоп\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет знака
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль окаймляющей линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер знака
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения знака мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool ArrowStopMove(const long   chart_ID=0,        // ID графика
                   const string name="ArrowStop", // имя объекта
                   datetime     time=0,           // координата времени точки привязки
                   double       price=0)          // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Меняет способ привязки знака "Стоп"                              |
//+------------------------------------------------------------------+
bool ArrowStopAnchorChange(const long              chart_ID=0,         // ID графика
                           const string            name="ArrowStop",   // имя объекта
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // положение точ
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим способ привязки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": не удалось изменить способ привязки! Код ошибки = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет знак "Стоп"                                              |
//+------------------------------------------------------------------+
bool ArrowStopDelete(const long   chart_ID=0,       // ID графика
                     const string name="ArrowStop") // имя знака
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим знак
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить знак \"Стоп\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точки привязки знака
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования знака
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- создадим знак "Стоп" на графике
   if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки и менять ее положение относительно знака
//--- счетчик цикла
   int h_steps=bars*2/5;
//--- перемещаем точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d<bars-1)
         d+=1;
      //--- сдвигаем точку
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.025 секунды
      Sleep(25);
     }
//--- меняем положение точки привязки относительно знака
   ArrowStopAnchorChange(0,InpName,ANCHOR_TOP);
//--- перерисуем график
   ChartRedraw();
//--- счетчик цикла
   h_steps=bars*2/5;
//--- перемещаем точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d<bars-1)
         d+=1;
      //--- сдвигаем точку
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.025 секунды
      Sleep(25);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим знак с графика
   ArrowStopDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_CHECK

Знак "Галка".



Примечание

Положение точки привязки относительно знака можно выбрать из перечисления ENUM_ARROW_ANCHOR.

Знаки большого размера (больше 5) можно создать только установив соответствующее значение свойства OBJPROP_WIDTH при написании кода в MetaEditor.

Пример

Следующий скрипт создает и перемещает на графике знак "Галка". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт рисует знак \"Галка\"."
#property description "Координата точки привязки задается в"
#property description "процентах от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string             InpName="ArrowCheck"; // Имя знака
input int                InpDate=10;           // Дата точки привязки в %
input int                InpPrice=50;          // Цена точки привязки в %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP; // Способ привязки
input color              InpColor=clrRed;      // Цвет знака
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Стиль окаймляющей линии
input int               InpWidth=5;           // Размер знака
input bool              InpBack=false;        // Знак на заднем плане
input bool              InpSelection=false;   // Выделить для перемещений
input bool              InpHidden=true;       // Скрыт в списке объектов
input long              InpZOrder=0;          // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает знак "Галка"                                             |
//+------------------------------------------------------------------+
bool ArrowCheckCreate(const long              chart_ID=0,           // ID графика
                      const string            name="ArrowCheck",    // имя знака
                      const int               sub_window=0,         // номер подокна
                      datetime                time=0,               // время точки при
                      double                  price=0,              // цена точки прив
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // способ привязки
                      const color             clr=clrRed,           // цвет знака
                      const ENUM_LINE_STYLE   style=STYLE_SOLID,    // стиль окаймляю
                      const int               width=3,              // размер знака
                      const bool              back=false,           // на заднем плане
                      const bool              selection=true,       // выделить для пе
                      const bool              hidden=true,          // скрыт в списке
                      const long              z_order=0)            // приоритет на на
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим знак
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать знак \"Галка\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет знака
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль окаймляющей линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер знака
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения знака мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установи приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool ArrowCheckMove(const long   chart_ID=0,         // ID графика
                    const string name="ArrowCheck", // имя объекта
                    datetime     time=0,            // координата времени точки привяз
                    double       price=0)           // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Меняет способ привязки "Галки"                                   |
//+------------------------------------------------------------------+
bool ArrowCheckAnchorChange(const long              chart_ID=0,        // ID графика
                            const string            name="ArrowCheck", // имя объекта
                            const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // способ привя
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим способ привязки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": не удалось изменить способ привязки! Код ошибки = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет знак "Галка"                                             |
//+------------------------------------------------------------------+
bool ArrowCheckDelete(const long   chart_ID=0,        // ID графика
                      const string name="ArrowCheck") // имя знака
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим знак
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить знак \"Галка\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точки привязки знака
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования знака
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- создадим знак "Галка" на графике
   if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки и менять ее положение относительно знака
//--- счетчик цикла
   int h_steps=bars*2/5;
//--- перемещаем точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d<bars-1)
         d+=1;
      //--- сдвигаем точку
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.025 секунды
      Sleep(25);
     }
//--- меняем положение точки привязки относительно знака
   ArrowCheckAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- перерисуем график
   ChartRedraw();
//--- счетчик цикла
   h_steps=bars*2/5;
//--- перемещаем точку привязки
   for(int i=0;i<h_steps;i++)
     {
      //--- возьмем следующее значение
      if(d<bars-1)
         d+=1;
      //--- сдвигаем точку
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.025 секунды
      Sleep(25);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим знак с графика
   ArrowCheckDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_LEFT_PRICE

Левая ценовая метка.



Пример

Следующий скрипт создает и перемещает на графике левую ценовую метку. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает левую ценовую метку на графике."
#property description "Координата точки привязки задается в процентах"
#property description "от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string            InpName="LeftPrice";  // Имя ценовой метки
input int               InpDate=100;          // Дата точки привязки в %
input int               InpPrice=10;          // Цена точки привязки в %
input color             InpColor=clrRed;      // Цвет ценовой метки
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Стиль окаймляющей линии
input int               InpWidth=2;           // Размер ценовой метки
input bool              InpBack=false;        // Метка на заднем плане
input bool              InpSelection=true;    // Выделить для перемещений
input bool              InpHidden=true;       // Скрыт в списке объектов
input long              InpZOrder=0;          // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает левую ценовую метку                                      |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowLeftPriceCreate(const long            chart_ID=0,        // ID графика
                          const string          name="LeftPrice",  // имя ценовой мет
                          const int             sub_window=0,      // номер подокна
                          datetime              time=0,            // время точки при
                          double                price=0,           // цена точки прив
                          const color           clr=clrRed,        // цвет ценовой ме
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль окаймляюще
                          const int             width=1,           // размер ценовой м
                          const bool            back=false,        // на заднем плане
                          const bool            selection=true,    // выделить для пер
                          const bool            hidden=true,       // скрыт в списке
                          const long            z_order=0)         // приоритет на на
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим ценовую метку
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать левую ценовую метку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет метки
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль окаймляющей линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер метки
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения метки мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ArrowLeftPriceMove(const long   chart_ID=0,        // ID графика
                        const string name="LeftPrice", // имя метки
                        datetime     time=0,           // координата времени точки при
                        double       price=0)          // координата цены точки привяз
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет левую ценовую метку с графика                            |
//+------------------------------------------------------------------+
bool ArrowLeftPriceDelete(const long   chart_ID=0,        // ID графика
                          const string name="LeftPrice") // имя метки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим метку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить левую ценовую метку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точки привязки метки
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования метки
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- создадим левую ценовую метку на графике
   if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки
//--- счетчик цикла
   int v_steps=accuracy*4/5;
//--- перемещаем точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p<accuracy-1)
         p+=1;
      //--- сдвигаем точку
      if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим метку с графика
   ArrowLeftPriceDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_RIGHT_PRICE

Правая ценовая метка.



Пример

Следующий скрипт создает и перемещает на графике правую ценовую метку. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает правую ценовую метку на графике."
#property description "Координата точки привязки задается в процентах"
#property description "от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string            InpName="RightPrice"; // Имя ценовой метки
input int               InpDate=0;            // Дата точки привязки в %
input int               InpPrice=90;          // Цена точки привязки в %
input color             InpColor=clrRed;      // Цвет ценовой метки
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Стиль окаймляющей линии
input int               InpWidth=2;           // Размер ценовой метки
input bool              InpBack=false;        // Метка на заднем плане
input bool              InpSelection=true;    // Выделить для перемещений
input bool              InpHidden=true;       // Скрыт в списке объектов
input long              InpZOrder=0;          // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает правую ценовую метку                                     |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowRightPriceCreate(const long            chart_ID=0,        // ID графика
                           const string          name="RightPrice", // имя ценовой мет
                           const int             sub_window=0,      // номер подокна
                           datetime              time=0,            // время точки при
                           double                price=0,           // цена точки прив
                           const color           clr=clrRed,        // цвет ценовой ме
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль окаймляю
                           const int             width=1,           // размер ценовой
                           const bool            back=false,        // на заднем плане
                           const bool            selection=true,    // выделить для пе
                           const bool            hidden=true,       // скрыт в списке
                           const long            z_order=0)         // приоритет на на
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим ценовую метку
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать правую ценовую метку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет метки
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль окаймляющей линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер метки
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения метки мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ArrowRightPriceMove(const long   chart_ID=0,        // ID графика
                         const string name="RightPrice", // имя метки
                         datetime     time=0,            // координата времени точки п
                         double       price=0)           // координата цены точки прив
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет правую ценовую метку с графика                           |
//+------------------------------------------------------------------+
bool ArrowRightPriceDelete(const long   chart_ID=0,        // ID графика
                           const string name="RightPrice") // имя метки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим метку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить правую ценовую метку! Код ошибки = ",GetLastError())
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точки привязки метки
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования метки
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- создадим правую ценовую метку на графике
   if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

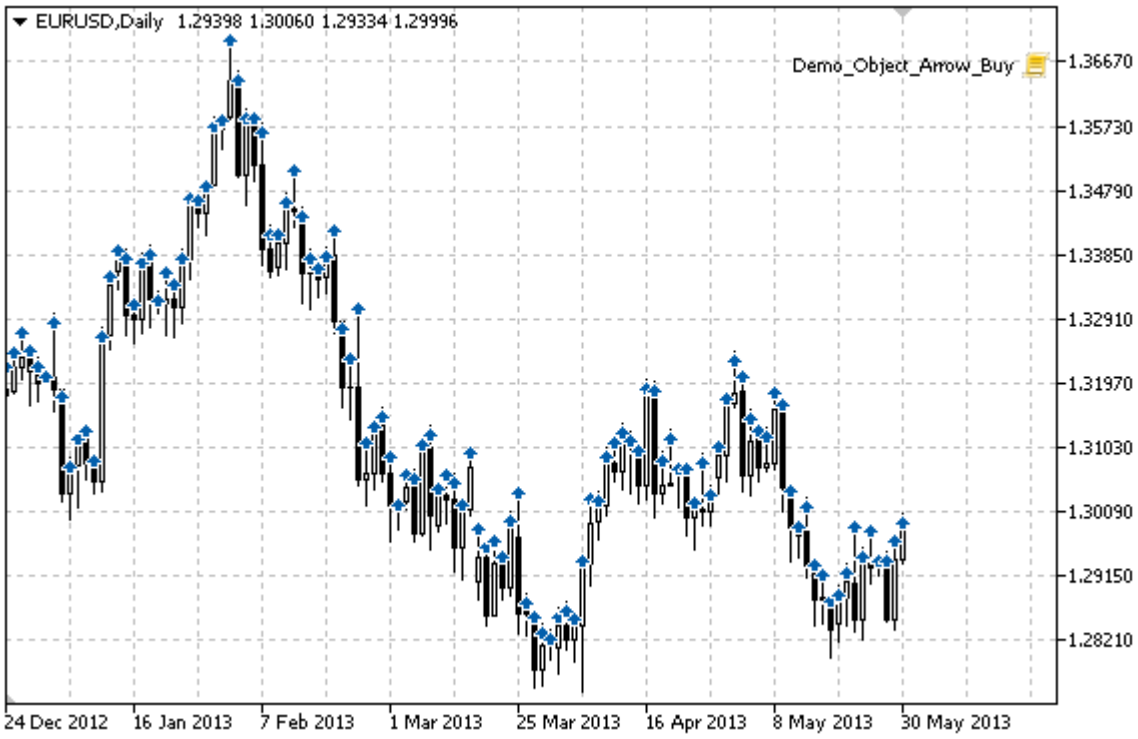
© 2000-2025, MetaQuotes Ltd.

```
      }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать точку привязки
//--- счетчик цикла
   int v_steps=accuracy*4/5;
//--- перемещаем точку привязки
   for(int i=0;i<v_steps;i++)
     {
      //--- возьмем следующее значение
      if(p>1)
         p-=1;
      //--- сдвигаем точку
      if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим метку с графика
   ArrowRightPriceDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_BUY

Знак "Buy".



Пример

Следующий скрипт создает и перемещает на графике значки "Buy". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт рисует значки \"Buy\" в окне графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input color InpColor=C'3,95,172'; // Цвет значков
//+------------------------------------------------------------------+
//| Создает знак "Buy"                                               |
//+------------------------------------------------------------------+
bool ArrowBuyCreate(const long            chart_ID=0,        // ID графика
                    const string          name="ArrowBuy",   // имя знака
                    const int             sub_window=0,      // номер подокна
                    datetime              time=0,            // время точки привязки
                    double                price=0,           // цена точки привязки
                    const color           clr=C'3,95,172',   // цвет знака
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии (при выдел
                    const int             width=1,           // размер линии (при выде
                    const bool            back=false,        // на заднем плане
                    const bool            selection=false,   // выделить для перемещен
```

© 2000-2025, MetaQuotes Ltd.

```
                    const bool            hidden=true,       // скрыт в списке объекто
                    const long            z_order=0)         // приоритет на нажатие м
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим знак
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать знак \"Buy\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет знака
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии (при выделении)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер линии (при выделении)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения знака мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установи приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool ArrowBuyMove(const long   chart_ID=0,       // ID графика
                  const string name="ArrowBuy", // имя объекта
                  datetime     time=0,          // координата времени точки привязки
                  double       price=0)         // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
       return(false);
      }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет знак "Buy"                                               |
//+------------------------------------------------------------------+
bool ArrowBuyDelete(const long   chart_ID=0,       // ID графика
                    const string name="ArrowBuy") // имя знака
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим знак
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить знак \"Buy\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // массив для хранения дат видимых баров
   double   low[];  // массив для хранения цен Low видимых баров
   double   high[]; // массив для хранения цен High видимых баров
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Не удалось скопировать значения цен Low! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("Не удалось скопировать значения цен High! Код ошибки = ",GetLastError());
      return;
     }
//--- создадим значки "Buy" в точке Low для каждого видимого бара
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- переместим значки "Buy" в точку High для каждого видимого бара
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- удалим значки "Buy"
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_SELL

Знак "Sell".



Пример

Следующий скрипт создает и перемещает на графике значки "Sell". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт рисует значки \"Sell\" в окне графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input color InpColor=C'225,68,29'; // Цвет значков
//+------------------------------------------------------------------+
//| Создает знак "Sell"                                              |
//+------------------------------------------------------------------+
bool ArrowSellCreate(const long            chart_ID=0,        // ID графика
                     const string          name="ArrowSell",  // имя знака
                     const int             sub_window=0,      // номер подокна
                     datetime              time=0,            // время точки привязки
                     double                price=0,           // цена точки привязки
                     const color           clr=C'225,68,29',  // цвет знака
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии (при выде
                     const int             width=1,           // размер линии (при выд
                     const bool            back=false,        // на заднем плане
                     const bool            selection=false,   // выделить для перемеще
```

© 2000-2025, MetaQuotes Ltd.

```
                     const bool            hidden=true,       // скрыт в списке объект
                     const long            z_order=0)         // приоритет на нажатие
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим знак
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать знак \"Sell\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим цвет знака
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии (при выделении)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер линии (при выделении)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения знака мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool ArrowSellMove(const long   chart_ID=0,        // ID графика
                   const string name="ArrowSell", // имя объекта
                   datetime     time=0,           // координата времени точки привязки
                   double       price=0)          // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
       return(false);
      }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет знак "Sell"                                              |
//+------------------------------------------------------------------+
bool ArrowSellDelete(const long   chart_ID=0,        // ID графика
                     const string name="ArrowSell") // имя знака
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим знак
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить знак \"Sell\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки, и для пустых значений         |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // массив для хранения дат видимых баров
   double   low[];  // массив для хранения цен Low видимых баров
   double   high[]; // массив для хранения цен High видимых баров
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Не удалось скопировать значения цен Low! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("Не удалось скопировать значения цен High! Код ошибки = ",GetLastError());
      return;
     }
//--- создадим значки "Sell" в точке High для каждого видимого бара
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- переместим значки "Sell" в точку Low для каждого видимого бара
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- удалим значки "Sell"
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW

Объект "Стрелка".



Примечание

Положение точки привязки относительно стрелки можно выбрать из перечисления ENUM_ARROW_ANCHOR.

Стрелки большого размера (больше 5) можно создать только установив соответствующее значение свойства OBJPROP_WIDTH при написании кода в MetaEditor.

Нужный тип стрелки можно выбрать, установив один из кодов символов шрифта Wingdings.

Пример

Следующий скрипт создает на графике объект "Стрелка" и изменяет ее тип. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает произвольную стрелку в окне графика."
#property description "Координата точки привязки задается в процентах"
#property description "от размеров окна графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string            InpName="Arrow";        // Имя стрелки
input int               InpDate=50;             // Дата точки привязки в %
input int               InpPrice=50;            // Цена точки привязки в %
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP;    // Способ привязки
input color             InpColor=clrDodgerBlue; // Цвет стрелки
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;   // Стиль окаймляющей линии
input int               InpWidth=10;            // Размер стрелки
input bool              InpBack=false;          // Стрелка на заднем плане
input bool              InpSelection=false;     // Выделить для перемещений
input bool              InpHidden=true;         // Скрыт в списке объектов
input long              InpZOrder=0;            // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает стрелку                                                  |
//+------------------------------------------------------------------+
bool ArrowCreate(const long              chart_ID=0,           // ID графика
                 const string            name="Arrow",         // имя стрелки
                 const int               sub_window=0,         // номер подокна
                 datetime                time=0,               // время точки привязки
                 double                  price=0,              // цена точки привязки
                 const uchar             arrow_code=252,       // код стрелки
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // положение точки при
                 const color             clr=clrRed,           // цвет стрелки
                 const ENUM_LINE_STYLE   style=STYLE_SOLID,    // стиль окаймляющей ли
                 const int               width=3,              // размер стрелки
                 const bool              back=false,           // на заднем плане
                 const bool              selection=true,       // выделить для перемещ
                 const bool              hidden=true,          // скрыт в списке объек
                 const long              z_order=0)            // приоритет на нажатие
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeArrowEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим стрелку
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать стрелку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим код стрелки
   ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
//--- установим способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет стрелки
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль окаймляющей линии
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер стрелки
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- включим (true) или отключим (false) режим перемещения стрелки мышью
//--- при создании графического объекта функцией ObjectCreate, по умолчанию объект
//--- нельзя выделить и перемещать. Внутри же этого метода параметр selection
//--- по умолчанию равен true, что позволяет выделять и перемещать этот объект
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool ArrowMove(const long   chart_ID=0,   // ID графика
               const string name="Arrow", // имя объекта
               datetime     time=0,       // координата времени точки привязки
               double       price=0)      // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Меняет код стрелки                                               |
//+------------------------------------------------------------------+
bool ArrowCodeChange(const long   chart_ID=0,   // ID графика
                     const string name="Arrow", // имя объекта
                     const uchar  code=252)     // код стрелки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим код стрелки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      Print(__FUNCTION__,
            ": не удалось изменить код стрелки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Меняет способ привязки                                           |
//+------------------------------------------------------------------+
bool ArrowAnchorChange(const long              chart_ID=0,        // ID графика
                       const string            name="Arrow",      // имя объекта
                       const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // способ привязки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим способ привязки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": не удалось изменить способ привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет стрелку                                                  |
//+------------------------------------------------------------------+
bool ArrowDelete(const long   chart_ID=0,   // ID графика
                 const string name="Arrow") // имя стрелки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим стрелку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить стрелку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- размер массива price
   int accuracy=1000;
//--- массивы для хранения значений дат и цен, которые будут использованы
//--- для установки и изменения координат точки привязки знака
   datetime date[];
   double   price[];
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен
//--- найдем максимальное и минимальное значение графика
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- определим шаг изменения цены и заполним массив
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- определим точки для рисования стрелки
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
```

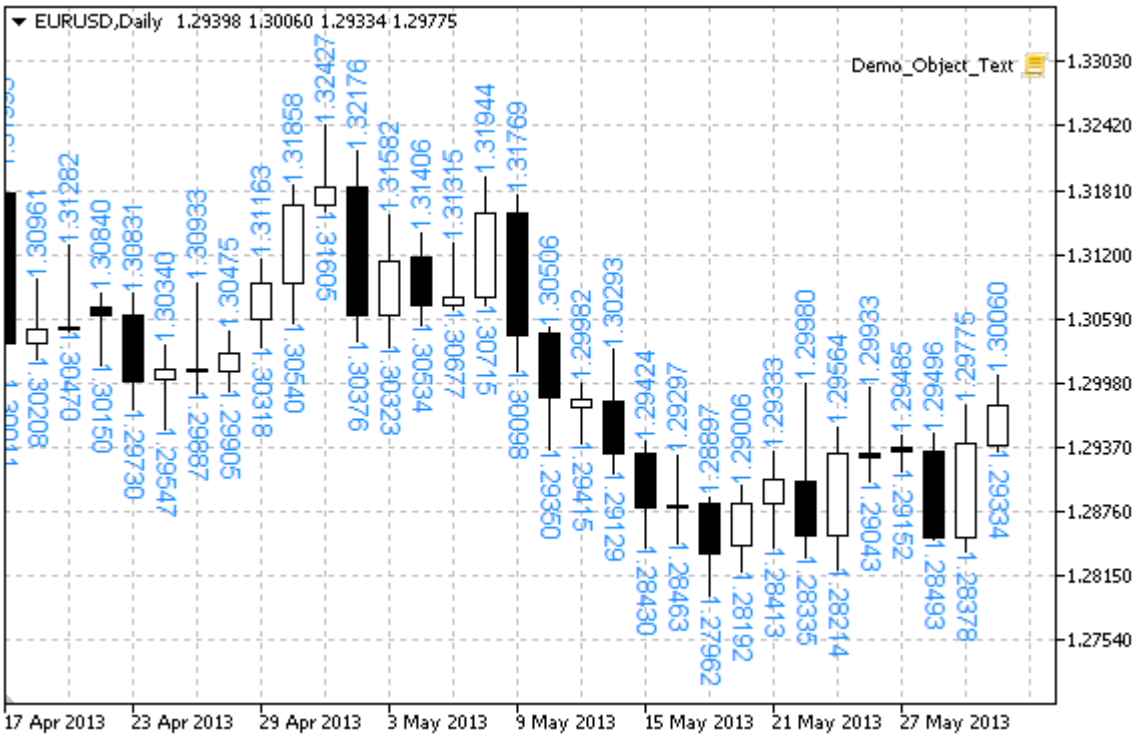
© 2000-2025, MetaQuotes Ltd.

```
//--- создадим стрелку на графике
   if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график
   ChartRedraw();
//--- в цикле рассмотрим все варианты построения стрелок
   for(int i=33;i<256;i++)
     {
      if(!ArrowCodeChange(0,InpName,(uchar)i))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в полсекунды
      Sleep(500);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим стрелку с графика
   ArrowDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TEXT

Объект "Текст".



Примечание

Положение точки привязки относительно текста можно выбрать из перечисления ENUM_ANCHOR_POINT. Также можно менять угол наклона текста при помощи свойства OBJPROP_ANGLE.

Пример

Следующий скрипт создает на графике несколько объектов "Текст". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает графический объект \"Текст\"."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string             InpFont="Arial";          // Шрифт
input int                InpFontSize=10;           // Размер шрифта
input color              InpColor=clrRed;          // Цвет
input double             InpAngle=90.0;            // Угол наклона в градусах
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_LEFT;    // Способ привязки
input bool               InpBack=false;            // Объект на заднем плане
input bool               InpSelection=false;       // Выделить для перемещений
input bool               InpHidden=true;           // Скрыт в списке объектов
```

© 2000-2025, MetaQuotes Ltd.

```
input long              InpZOrder=0;              // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает объект "Текст"                                           |
//+------------------------------------------------------------------+
bool TextCreate(const long              chart_ID=0,               // ID графика
                const string            name="Text",              // имя объекта
                const int               sub_window=0,             // номер подокна
                datetime                time=0,                   // время точки привя
                double                  price=0,                  // цена точки привяз
                const string            text="Text",              // сам текст
                const string            font="Arial",             // шрифт
                const int               font_size=10,             // размер шрифта
                const color             clr=clrRed,               // цвет
                const double            angle=0.0,                // наклон текста
                const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // способ привязки
                const bool              back=false,               // на заднем плане
                const bool              selection=false,          // выделить для пере
                const bool              hidden=true,              // скрыт в списке об
                const long              z_order=0)                // приоритет на нажа
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeTextEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим объект "Текст"
   if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать объект \"Текст\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим текст
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- установим шрифт текста
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- установим размер шрифта
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- установим угол наклона текста
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- установим способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения объекта мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает точку привязки                                        |
//+------------------------------------------------------------------+
bool TextMove(const long   chart_ID=0,  // ID графика
              const string name="Text", // имя объекта
              datetime     time=0,      // координата времени точки привязки
              double       price=0)     // координата цены точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет текст объекта                                           |
//+------------------------------------------------------------------+
bool TextChange(const long   chart_ID=0,  // ID графика
                const string name="Text", // имя объекта
                const string text="Text") // текст
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим текст объекта
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": не удалось изменить текст! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Удаляет объект "Текст"                                           |
//+------------------------------------------------------------------+
bool TextDelete(const long   chart_ID=0,  // ID графика
                const string name="Text") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим объект
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить объект \"Текст\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeTextEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // массив для хранения дат видимых баров
   double   low[];  // массив для хранения цен Low видимых баров
   double   high[]; // массив для хранения цен High видимых баров
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
       return;
      }
//--- заполним массив цен Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
      {
       Print("Не удалось скопировать значения цен Low! Код ошибки = ",GetLastError());
       return;
      }
//--- заполним массив цен High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
      {
       Print("Не удалось скопировать значения цен High! Код ошибки = ",GetLastError());
       return;
      }
//--- определим, как часто надо делать надписи
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- определим шаг
   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;
      }
//--- создадим надписи для значений High и Low баров (с промежутками)
   for(int i=0;i<bars;i+=step)
      {
       //--- создаем надписи
       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;
         }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в полсекунды
   Sleep(500);
//--- удалим надписи
   for(int i=0;i<bars;i+=step)
     {
      if(!TextDelete(0,"TextHigh_"+(string)i))
         return;
      if(!TextDelete(0,"TextLow_"+(string)i))
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_LABEL

Объект "Текстовая метка".



Примечание

Положение точки привязки относительно метки можно выбрать из перечисления ENUM_ANCHOR_POINT. Координаты точки привязки задаются в пикселях.

Также можно выбрать угол привязки текстовой метки из перечисления ENUM_BASE_CORNER.

Пример

Следующий скрипт создает и перемещает на графике объект "Текстовая метка". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает графический объект \"Текстовая метка\"."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string             InpName="Label";          // Имя метки
input int                InpX=150;                 // Расстояние по оси X
input int                InpY=150;                 // Расстояние по оси Y
input string             InpFont="Arial";          // Шрифт
input int                InpFontSize=14;           // Размер шрифта
input color              InpColor=clrRed;          // Цвет
input double             InpAngle=0.0;             // Угол наклона в градусах
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // Способ привязки
```

© 2000-2025, MetaQuotes Ltd.

```
input bool               InpBack=false;           // Объект на заднем плане
input bool               InpSelection=true;       // Выделить для перемещений
input bool               InpHidden=true;          // Скрыт в списке объектов
input long               InpZOrder=0;             // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает текстовую метку                                          |
//+------------------------------------------------------------------+
bool LabelCreate(const long              chart_ID=0,               // ID графика
                 const string            name="Label",             // имя метки
                 const int               sub_window=0,             // номер подокна
                 const int               x=0,                      // координата по ос
                 const int               y=0,                      // координата по ос
                 const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // угол графика для
                 const string            text="Label",             // текст
                 const string            font="Arial",             // шрифт
                 const int               font_size=10,             // размер шрифта
                 const color             clr=clrRed,               // цвет
                 const double            angle=0.0,                // наклон текста
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // способ привязки
                 const bool              back=false,               // на заднем плане
                 const bool              selection=false,          // выделить для пер
                 const bool              hidden=true,              // скрыт в списке с
                 const long              z_order=0)                // приоритет на нах
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим текстовую метку
   if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать текстовую метку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим координаты метки
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- установим угол графика, относительно которого будут определяться координаты точк
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- установим текст
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- установим шрифт текста
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- установим размер шрифта
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- установим угол наклона текста
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- установим способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения метки мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает текстовую метку                                       |
//+------------------------------------------------------------------+
bool LabelMove(const long   chart_ID=0,   // ID графика
               const string name="Label", // имя метки
               const int    x=0,          // координата по оси X
               const int    y=0)          // координата по оси Y
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим текстовую метку
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": не удалось переместить X-координату метки! Код ошибки = ",GetLastError
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": не удалось переместить Y-координату метки! Код ошибки = ",GetLastError
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет угол графика для привязки метки                         |
//+------------------------------------------------------------------+
bool LabelChangeCorner(const long             chart_ID=0,               // ID графика
                       const string           name="Label",             // имя метки
                       const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // угол графи
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим угол привязки
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": не удалось изменить угол привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет текст метки                                             |
//+------------------------------------------------------------------+
bool LabelTextChange(const long   chart_ID=0,   // ID графика
                     const string name="Label", // имя объекта
                     const string text="Text")  // текст
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим текст объекта
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": не удалось изменить текст! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет текстовую метку                                          |
//+------------------------------------------------------------------+
bool LabelDelete(const long   chart_ID=0,   // ID графика
                 const string name="Label") // имя метки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим метку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить текстовую метку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- запомним координаты метки в локальные переменные
   int x=InpX;
   int y=InpY;
//--- размеры окна графика
   long x_distance;
   long y_distance;
//--- определим размеры окна
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Не удалось получить ширину графика! Код ошибки = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Не удалось получить высоту графика! Код ошибки = ",GetLastError());
      return;
     }
//--- проверим входные параметры на корректность
   if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- подготовим начальный текст для метки
   string text;
   StringConcatenate(text,"Левый верхний угол: ",x,",",y);
//--- создадим текстовую метку на графике
   if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
      InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем полсекунды
   ChartRedraw();
   Sleep(500);
//--- будем перемещать метку и одновременно менять ее текст
//--- количество итераций по осям
   int h_steps=(int)(x_distance/2-InpX);
   int v_steps=(int)(y_distance/2-InpY);
//--- переместим метку вниз
   for(int i=0;i<v_steps;i++)
     {
      //--- меняем координату
      y+=2;
      //--- перемещаем метку и меняем ее текст
      MoveAndTextChange(x,y,"Левый верхний угол: ");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- задержка в полсекунды
   Sleep(500);
//--- переместим метку вправо
   for(int i=0;i<h_steps;i++)
     {
      //--- меняем координату
      x+=2;
      //--- перемещаем метку и меняем ее текст
      MoveAndTextChange(x,y,"Левый верхний угол: ");
     }
//--- задержка в полсекунды
   Sleep(500);
//--- переместим метку вверх
   for(int i=0;i<v_steps;i++)
     {
      //--- меняем координату
      y-=2;
      //--- перемещаем метку и меняем ее текст
      MoveAndTextChange(x,y,"Левый верхний угол: ");
     }
//--- задержка в полсекунды
   Sleep(500);
//--- переместим метку влево
   for(int i=0;i<h_steps;i++)
     {
      //--- меняем координату
      x-=2;
      //--- перемещаем метку и меняем ее текст
      MoveAndTextChange(x,y,"Левый верхний угол: ");
     }
//--- задержка в полсекунды
   Sleep(500);
//--- теперь переместим точку путем изменения угла привязки
//--- переместим в левый нижний угол
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
      return;
//--- изменим текст метки
   StringConcatenate(text,"Левый нижний угол: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- перерисуем график и подождем две секунды
   ChartRedraw();
   Sleep(2000);
//--- переместим в правый нижний угол
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
      return;
//--- изменим текст метки
   StringConcatenate(text,"Правый нижний угол: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
//--- перерисуем график и подождем две секунды
   ChartRedraw();
   Sleep(2000);
//--- переместим в правый верхний угол
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_UPPER))
      return;
//--- изменим текст метки
   StringConcatenate(text,"Правый верхний угол: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- перерисуем график и подождем две секунды
   ChartRedraw();
   Sleep(2000);
//--- переместим в левый верхний угол
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_UPPER))
      return;
//--- изменим текст метки
   StringConcatenate(text,"Левый верхний угол: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- перерисуем график и подождем две секунды
   ChartRedraw();
   Sleep(2000);
//--- удалим метку
   LabelDelete(0,InpName);
//--- перерисуем график и подождем полсекунды
   ChartRedraw();
   Sleep(500);
//---
  }
//+------------------------------------------------------------------+
//| Функция перемещает объект и меняет его текст                     |
//+------------------------------------------------------------------+
bool MoveAndTextChange(const int x,const int y,string text)
  {
//--- перемещаем метку
   if(!LabelMove(0,InpName,x,y))
      return(false);
//--- изменим текст метки
   StringConcatenate(text,text,x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return(false);
//--- проверим факт принудительного завершения скрипта
   if(IsStopped())
      return(false);
//--- перерисуем график
   ChartRedraw();
// задержка в 0.01 секунды
```

© 2000-2025, MetaQuotes Ltd.

```
    Sleep(10);
//--- выход из функции
    return(true);
   }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BUTTON

Объект "Кнопка".



Примечание

Координаты точки привязки задаются в пикселях. Можно выбрать угол привязки кнопки из перечисления ENUM_BASE_CORNER.

Пример

Следующий скрипт создает и перемещает на графике объект "Кнопка". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает кнопку на графике."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="Button";           // Имя кнопки
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Угол графика для привязки
input string           InpFont="Arial";            // Шрифт
input int              InpFontSize=14;             // Размер шрифта
input color            InpColor=clrBlack;          // Цвет текста
input color            InpBackColor=C'236,233,216'; // Цвет фона
input color            InpBorderColor=clrNONE;     // Цвет границы
input bool             InpState=false;             // Нажата/Отжата
input bool             InpBack=false;              // Объект на заднем плане
input bool             InpSelection=false;         // Выделить для перемещений
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpHidden=true;              // Скрыт в списке объектов
input long              InpZOrder=0;                 // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает кнопку                                                   |
//+------------------------------------------------------------------+
bool ButtonCreate(const long              chart_ID=0,               // ID графика
                  const string            name="Button",            // имя кнопки
                  const int               sub_window=0,             // номер подокна
                  const int               x=0,                      // координата по
                  const int               y=0,                      // координата по
                  const int               width=50,                 // ширина кнопки
                  const int               height=18,                // высота кнопки
                  const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // угол графика дл
                  const string            text="Button",            // текст
                  const string            font="Arial",             // шрифт
                  const int               font_size=10,             // размер шрифта
                  const color             clr=clrBlack,             // цвет текста
                  const color             back_clr=C'236,233,216',  // цвет фона
                  const color             border_clr=clrNONE,       // цвет границы
                  const bool              state=false,              // нажата/отжата
                  const bool              back=false,               // на заднем плане
                  const bool              selection=false,          // выделить для пе
                  const bool              hidden=true,              // скрыт в списке
                  const long              z_order=0)                // приоритет на на
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим кнопку
   if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать кнопку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим координаты кнопки
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- установим размер кнопки
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- установим угол графика, относительно которого будут определяться координаты точк
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- установим текст
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- установим шрифт текста
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- установим размер шрифта
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- установим цвет текста
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим цвет фона
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- установим цвет границы
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- переведем кнопку в заданное состояние
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- включим (true) или отключим (false) режим перемещения кнопки мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает кнопку                                                |
//+------------------------------------------------------------------+
bool ButtonMove(const long   chart_ID=0,    // ID графика
                const string name="Button", // имя кнопки
                const int    x=0,           // координата по оси X
                const int    y=0)           // координата по оси Y
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим кнопку
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": не удалось переместить X-координату кнопки! Код ошибки = ",GetLastError
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": не удалось переместить Y-координату кнопки! Код ошибки = ",GetLastError
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет размер кнопки                                           |
//+------------------------------------------------------------------+
bool ButtonChangeSize(const long   chart_ID=0,    // ID графика
```

© 2000-2025, MetaQuotes Ltd.

```
                      const string name="Button", // имя кнопки
                      const int    width=50,      // ширина кнопки
                      const int    height=18)     // высота кнопки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим размеры кнопки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": не удалось изменить ширину кнопки! Код ошибки = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": не удалось изменить высоту кнопки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет угол графика для привязки кнопки                        |
//+------------------------------------------------------------------+
bool ButtonChangeCorner(const long             chart_ID=0,               // ID графика
                        const string           name="Button",            // имя кнопки
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // угол графи
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим угол привязки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": не удалось изменить угол привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет текст кнопки                                            |
//+------------------------------------------------------------------+
bool ButtonTextChange(const long   chart_ID=0,    // ID графика
                      const string name="Button", // имя кнопки
                      const string text="Text")   // текст
  {
//--- сбросим значение ошибки
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- изменим текст объекта
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": не удалось изменить текст! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет кнопку                                                   |
//+------------------------------------------------------------------+
bool ButtonDelete(const long   chart_ID=0,    // ID графика
                  const string name="Button") // имя кнопки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим кнопку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить кнопку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- размеры окна графика
   long x_distance;
   long y_distance;
//--- определим размеры окна
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Не удалось получить ширину графика! Код ошибки = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Не удалось получить высоту графика! Код ошибки = ",GetLastError());
      return;
     }
//--- определим шаг для изменения размеров кнопки
```

© 2000-2025, MetaQuotes Ltd.

```
   int x_step=(int)x_distance/32;
   int y_step=(int)y_distance/32;
//--- установим координаты кнопки и ее размер
   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;
//--- создадим кнопку
   if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner,"Press",InpFont,InpFontSiz
      InpColor,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- перерисуем график
   ChartRedraw();
//--- в цикле уменьшаем кнопку
   int i=0;
   while(i<13)
     {
      //--- задержка в полсекунды
      Sleep(500);
      //--- переведем кнопку в нажатое состояние
      ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
      //--- перерисуем график и подождем 0.2 секунды
      ChartRedraw();
      Sleep(200);
      //--- переопределим координаты и размер кнопки
      x+=x_step;
      y+=y_step;
      x_size-=x_step*2;
      y_size-=y_step*2;
      //--- уменьшим кнопку
      ButtonMove(0,InpName,x,y);
      ButtonChangeSize(0,InpName,x_size,y_size);
      //--- вернем кнопку в ненажатое состояние
      ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
      //--- перерисуем график
      ChartRedraw();
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- увеличим счетчик цикла
      i++;
     }
//--- задержка в полсекунды
   Sleep(500);
//--- удалим кнопку
   ButtonDelete(0,InpName);
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- подождем 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_CHART

Объект "График".



Примечание

Объект типа "OBJ_CHART" не поддерживается (не отображается) при визуальном тестировании.

Координаты точки привязки задаются в пикселях. Можно выбрать угол привязки из перечисления ENUM_BASE_CORNER.

Для объекта "График" можно выбирать символ, период и масштаб, а также включать/отключать режим отображения шкалы цены и даты.

Пример

Следующий скрипт создает и перемещает на графике объект "График". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает объект \"График\"."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="Chart";              // Имя объекта
input string           InpSymbol="EURUSD";           // Символ
input ENUM_TIMEFRAMES  InpPeriod=PERIOD_H1;          // Период
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Угол для привязки
input int              InpScale=2;                   // Масштаб
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpDateScale=true;           // Отображение шкалы времени
input bool              InpPriceScale=true;          // Отображение шкалы цены
input color             InpColor=clrRed;             // Цвет рамки при выделении
input ENUM_LINE_STYLE   InpStyle=STYLE_DASHDOTDOT;   // Стиль линии при выделении
input int               InpPointWidth=1;             // Размер точки для перемещений
input bool              InpBack=false;               // Объект на заднем плане
input bool              InpSelection=true;           // Выделить для перемещений
input bool              InpHidden=true;              // Скрыт в списке объектов
input long              InpZOrder=0;                 // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает объект "График"                                          |
//+------------------------------------------------------------------+
bool ObjectChartCreate(const long              chart_ID=0,               // ID графика
                       const string            name="Chart",             // имя объект
                       const int               sub_window=0,             // номер подо
                       const string            symbol="EURUSD",          // символ
                       const ENUM_TIMEFRAMES   period=PERIOD_H1,         // период
                       const int               x=0,                      // координата
                       const int               y=0,                      // координата
                       const int               width=300,                // ширина
                       const int               height=200,               // высота
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // угол для п
                       const int               scale=2,                  // масштаб
                       const bool              date_scale=true,          // отображени
                       const bool              price_scale=true,         // отображени
                       const color             clr=clrRed,               // цвет рамки
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // стиль лини
                       const int               point_width=1,            // размер точ
                       const bool              back=false,               // на заднем
                       const bool              selection=false,          // выделить д
                       const bool              hidden=true,              // скрыт в сп
                       const long              z_order=0)                // приоритет
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим объект "График"
   if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать объект \"График\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим координаты объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- установим размер объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- установим угол графика, относительно которого будут определяться координаты точк
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- установим символ
   ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//--- установим период
   ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- установим масштаб
   ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- отобразим (true) или скроем (false) шкалу времени
   ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//--- отобразим (true) или скроем (false) шкалу цены
   ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- установим цвет рамки при включенном режиме выделения объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии рамки при включенном режиме выделения объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер точки привязки, с помощью которой можно перемещать объект
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения метки мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Устанавливает символ и таймфрейм объекта "График"                |
//+------------------------------------------------------------------+
bool ObjectChartSetSymbolAndPeriod(const long            chart_ID=0,        // ID графи
                                   const string          name="Chart",      // имя объе
                                   const string          symbol="EURUSD",   // символ
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // таймфрей
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим символ и таймфрейм объекта "График"
   if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
     {
      Print(__FUNCTION__,
            ": не удалось установить символ для объекта \"График\"! Код ошибки = ",Get
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": не удалось установить период для объекта \"График\"! Код ошибки = ",Get
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает объект "График"                                       |
//+------------------------------------------------------------------+
bool ObjectChartMove(const long   chart_ID=0,   // ID графика (не объекта)
                     const string name="Chart", // имя объекта
                     const int    x=0,          // координата по оси X
                     const int    y=0)          // координата по оси Y
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим объект
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": не удалось переместить X-координату объекта \"График\"! Код ошибки = ",
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": не удалось переместить Y-координату объекта \"График\"! Код ошибки = ",
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет размер объекта "График"                                 |
//+------------------------------------------------------------------+
bool ObjectChartChangeSize(const long   chart_ID=0,   // ID графика (не объекта)
                           const string name="Chart", // имя объекта
                           const int    width=300,    // ширина
                           const int    height=200)   // высота
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим размеры объекта
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": не удалось изменить ширину объекта \"График\"! Код ошибки = ",GetLastEr
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": не удалось изменить высоту объекта \"График\"! Код ошибки = ",GetLastEr
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Возвращает ID объекта "График"                                   |
//+------------------------------------------------------------------+
long ObjectChartGetID(const long   chart_ID=0,   // ID графика (не объекта)
                      const string name="Chart") // имя объекта
  {
//--- подготовим переменную для получения ID объекта "График"
   long id=-1;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим ID
   if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
     {
      Print(__FUNCTION__,
            ": не удалось получить ID объекта \"График\"! Код ошибки = ",GetLastError
     }
//--- возврат результата
   return(id);
  }
//+------------------------------------------------------------------+
//| Удаляет объект "График"                                          |
//+------------------------------------------------------------------+
bool ObjectChartDelete(const long   chart_ID=0,   // ID графика (не объекта)
                       const string name="Chart") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим кнопку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить объект \"График\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- получим количество символов в "Обзоре рынка"
   int  symbols=SymbolsTotal(true);
//--- определим, есть ли символ с указанным именем в списке символов
   bool exist=false;
   for(int i=0;i<symbols;i++)
      if(InpSymbol==SymbolName(i,true))
        {
         exist=true;
         break;
        }
   if(!exist)
     {
      Print("Ошибка! Данный символ ",InpSymbol," не представлен в окне \"Обзор Рынка\"
      return;
     }
//--- проверка входных параметров на корректность
   if(InpScale<0 || InpScale>5)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }

//--- размеры окна графика
   long x_distance;
   long y_distance;
//--- определим размеры окна
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Не удалось получить ширину графика! Код ошибки = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Не удалось получить высоту графика! Код ошибки = ",GetLastError());
      return;
     }
//--- установим координаты объекта "График" и его размер
   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;
//--- создадим объект "График"
   if(!ObjectChartCreate(0,InpName,0,InpSymbol,InpPeriod,x,y,x_size,y_size,InpCorner,I
      InpPriceScale,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
```

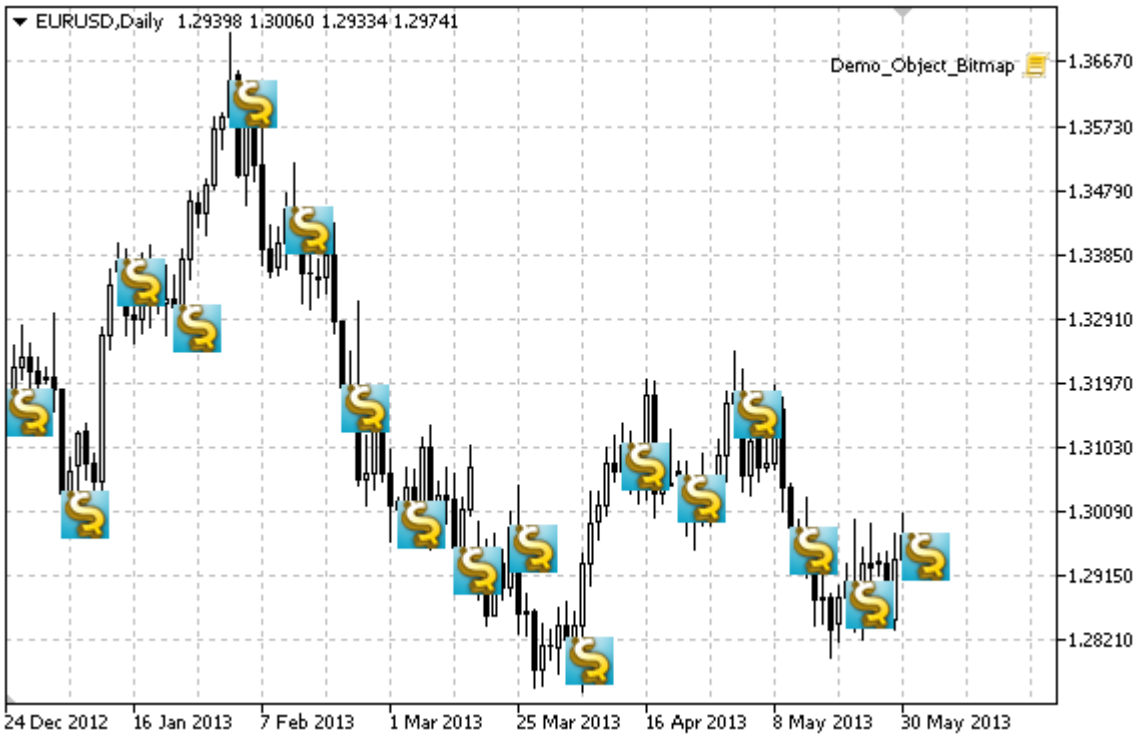
© 2000-2025, MetaQuotes Ltd.

```
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- растянем объект "График"
   int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
   for(int i=0;i<steps;i++)
     {
      //--- изменим размеры
      x_size+=1;
      y_size+=1;
      if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график и подождем 0.01 секунды
      ChartRedraw();
      Sleep(10);
     }
//--- задержка в полсекунды
   Sleep(500);
//--- изменим таймфрейм графика
   if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
      return;
   ChartRedraw();
//--- задержка в три секунды
   Sleep(3000);
//--- удалим объект
   ObjectChartDelete(0,InpName);
   ChartRedraw();
//--- подождем 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BITMAP

Объект "Рисунок".



Примечание

Для объекта "Рисунок" можно выбирать область видимости рисунка.

Пример

Следующий скрипт создает на графике несколько картинок. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает рисунок в окне графика."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string          InpFile="\\Images\\dollar.bmp"; // Имя файла с картинкой
input int             InpWidth=24;                    // X-координата области видимост
input int             InpHeight=24;                   // Y-координата области видимост
input int             InpXOffset=4;                   // Смещение области видимости по
input int             InpYOffset=4;                   // Смещение области видимости по
input color           InpColor=clrRed;                // Цвет рамки при выделении
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;           // Стиль линии при выделении
input int             InpPointWidth=1;                // Размер точки для перемещений
input bool            InpBack=false;                  // Объект на заднем плане
input bool            InpSelection=false;             // Выделить для перемещений
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpHidden=true;                  // Скрыт в списке объектов
input long            InpZOrder=0;                     // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает рисунок в окне графика                                   |
//+------------------------------------------------------------------+
bool BitmapCreate(const long            chart_ID=0,        // ID графика
                  const string          name="Bitmap",     // имя рисунка
                  const int             sub_window=0,      // номер подокна
                  datetime              time=0,            // время точки привязки
                  double                price=0,           // цена точки привязки
                  const string          file="",           // имя файла с картинкой
                  const int             width=10,          // X-координата области вид
                  const int             height=10,         // Y-координата области вид
                  const int             x_offset=0,        // смещение области видимос
                  const int             y_offset=0,        // смещение области видимос
                  const color           clr=clrRed,        // цвет рамки при выделении
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // стиль линии при выделени
                  const int             point_width=1,     // размер точки перемещения
                  const bool            back=false,        // на заднем плане
                  const bool            selection=false,   // выделить для перемещений
                  const bool            hidden=true,       // скрыт в списке объектов
                  const long            z_order=0)         // приоритет на нажатие мы
  {
//--- установим координаты точки привязки, если они не заданы
   ChangeBitmapEmptyPoint(time,price);
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим рисунок
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось создать рисунок в окне графика! Код ошибки = ",GetLastError
      return(false);
     }
//--- установим путь к файлу с картинкой
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": не удалось загрузить картинку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим область видимости изображения; если значения ширины или высоты
//--- больше значений ширины и высоты (соответственно) исходного изображения, то
//--- оно не рисуется; если значения ширины и высоты меньше размеров изображения,
//--- то рисуется та его часть, которая соответствует этим размерам
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- установим, какая часть изображения должна показываться в области видимости
//--- по умолчанию это левая верхняя область изображения; значения позволяют
```

© 2000-2025, MetaQuotes Ltd.

```
//--- произвести сдвиг от этого угла и отобразить другую часть изображения
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- установим цвет рамки при включенном режиме выделения объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии рамки при включенном режиме выделения объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер точки привязки, с помощью которой можно перемещать объект
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения метки мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Устанавливает новую картинку для рисунка                         |
//+------------------------------------------------------------------+
bool BitmapSetImage(const long   chart_ID=0,     // ID графика
                    const string name="Bitmap", // имя рисунка
                    const string file="")       // путь к файлу
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим путь к файлу с картинкой
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": не удалось загрузить картинку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает рисунок в окне графика                                |
//+------------------------------------------------------------------+
bool BitmapMove(const long   chart_ID=0,     // ID графика
                const string name="Bitmap", // имя рисунка
                datetime     time=0,        // время точки привязки
                double       price=0)       // цена точки привязки
  {
//--- если координаты точки не заданы, то перемещаем ее на текущий бар с ценой Bid
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим точку привязки
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": не удалось переместить точку привязки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет размер области видимости (размер рисунка)               |
//+------------------------------------------------------------------+
bool BitmapChangeSize(const long   chart_ID=0,     // ID графика
                      const string name="Bitmap", // имя рисунка
                      const int    width=0,       // ширина рисунка
                      const int    height=0)      // высота рисунка
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим размеры рисунка
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": не удалось изменить ширину рисунка! Код ошибки = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": не удалось изменить высоту рисунка! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет координату левого верхнего угла области видимости       |
//+------------------------------------------------------------------+
bool BitmapMoveVisibleArea(const long   chart_ID=0,     // ID графика
                           const string name="Bitmap", // имя рисунка
                           const int    x_offset=0,    // координата X области видимос
                           const int    y_offset=0)    // координата Y области видимос
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим координаты области видимости рисунка
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": не удалось изменить X-координату области видимости! Код ошибки = ",GetL
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": не удалось изменить Y-координату области видимости! Код ошибки = ",GetL
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет рисунок                                                  |
//+------------------------------------------------------------------+
bool BitmapDelete(const long   chart_ID=0,     // ID графика
                  const string name="Bitmap") // имя рисунка
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим метку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить рисунок! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Проверяет значения точки привязки и для пустых значений          |
//| устанавливает значения по умолчанию                              |
//+------------------------------------------------------------------+
void ChangeBitmapEmptyPoint(datetime &time,double &price)
  {
//--- если время точки не задано, то она будет на текущем баре
   if(!time)
      time=TimeCurrent();
//--- если цена точки не задана, то она будет иметь значение Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[];  // массив для хранения дат видимых баров
   double   close[]; // массив для хранения цен Close
//--- имя файла с картинкой
   string   file="\\Images\\dollar.bmp";
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- выделение памяти
   ArrayResize(date,bars);
   ArrayResize(close,bars);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- заполним массив цен Close
   if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
     {
      Print("Не удалось скопировать значения цен Close! Код ошибки = ",GetLastError())
      return;
     }
//--- определим, как часто надо выводить картинки
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- определим шаг
   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;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- создадим рисунки для значений High и Low баров (с промежутками)
   for(int i=0;i<bars;i+=step)
     {
      //--- создаем рисунки
      if(!BitmapCreate(0,"Bitmap_"+(string)i,0,date[i],close[i],InpFile,InpWidth,InpHe
         InpYOffset,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
        {
         return;
        }
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в полсекунды
   Sleep(500);
//--- удалим значки "Sell"
   for(int i=0;i<bars;i+=step)
     {
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BITMAP_LABEL

Объект "Графическая метка".



Примечание

Положение точки привязки относительно метки можно выбрать из перечисления ENUM_ANCHOR_POINT. Координаты точки привязки задаются в пикселях.

Также можно выбрать угол привязки графической метки из перечисления ENUM_BASE_CORNER.

Для объекта "Графическая метка" можно выбирать область видимости рисунка.

Пример

Следующий скрипт создает на графике несколько картинок. Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает объект \"Графическая метка\"."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string             InpName="BmpLabel";               // Имя метки
input string             InpFileOn="\\Images\\dollar.bmp"; // Имя файла для режима On
input string             InpFileOff="\\Images\\euro.bmp";  // Имя файла для режима Off
input bool               InpState=false;                   // Метка нажата/отжата
input ENUM_BASE_CORNER   InpCorner=CORNER_LEFT_UPPER;      // Угол графика для привязки
input ENUM_ANCHOR_POINT  InpAnchor=ANCHOR_CENTER;          // Способ привязки
input color              InpColor=clrRed;                  // Цвет рамки при выделении
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE    InpStyle=STYLE_SOLID;              // Стиль линии при выделени
input int                InpPointWidth=1;                   // Размер точки для перемеще
input bool               InpBack=false;                     // Объект на заднем плане
input bool               InpSelection=false;                // Выделить для перемещений
input bool               InpHidden=true;                    // Скрыт в списке объектов
input long               InpZOrder=0;                       // Приоритет на нажатие мыш
//+------------------------------------------------------------------+
//| Создает объект "Графическая метка"                               |
//+------------------------------------------------------------------+
bool BitmapLabelCreate(const long              chart_ID=0,               // ID графика
                       const string            name="BmpLabel",          // имя метки
                       const int               sub_window=0,             // номер подо
                       const int               x=0,                      // координата
                       const int               y=0,                      // координата
                       const string            file_on="",               // картинка н
                       const string            file_off="",              // картинка н
                       const int               width=0,                  // X-координа
                       const int               height=0,                 // Y-координа
                       const int               x_offset=10,              // смещение с
                       const int               y_offset=10,              // смещение с
                       const bool              state=false,              // нажата/отж
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // угол графи
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // способ при
                       const color             clr=clrRed,               // цвет рамки
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // стиль лини
                       const int               point_width=1,            // размер точ
                       const bool              back=false,               // на заднем
                       const bool              selection=false,          // выделить д
                       const bool              hidden=true,              // скрыт в сп
                       const long              z_order=0)                // приоритет
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим графическую метку
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать объект \"Графическая метка\"! Код ошибки = ",GetLast
      return(false);
     }
//--- установим картинки для режимов On и Off
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
     {
      Print(__FUNCTION__,
            ": не удалось загрузить картинку для режима On! Код ошибки = ",GetLastErro
      return(false);
     }
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": не удалось загрузить картинку для режима Off! Код ошибки = ",GetLastErr
      return(false);
     }
//--- установим координаты метки
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- установим область видимости изображения; если значения ширины или высоты
//--- больше значений ширины и высоты (соответственно) исходного изображения, то
//--- оно не рисуется; если значения ширины и высоты меньше размеров изображения,
//--- то рисуется та его часть, которая соответствует этим размерам
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- установим, какая часть изображения должна показываться в области видимости
//--- по умолчанию это левая верхняя область изображения; значения позволяют
//--- произвести сдвиг от этого угла и отобразить другую часть изображения
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- установим, в каком сотоянии находится метка (нажатом или отжатом)
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- установим угол графика, относительно которого будут определяться координаты точк
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- установим способ привязки
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- установим цвет рамки при включенном режиме выделения объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии рамки при включенном режиме выделения объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим размер точки привязки, с помощью которой можно перемещать объект
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения метки мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Устанавливает новую картинку для объекта "Графическая метка"     |
//+------------------------------------------------------------------+
bool BitmapLabelSetImage(const long   chart_ID=0,      // ID графика
                         const string name="BmpLabel", // имя метки
                         const int    on_off=0,        // модификатор (On или Off)
                         const string file="")         // путь к файлу
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим путь к файлу с картинкой
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
     {
      Print(__FUNCTION__,
            ": не удалось загрузить картинку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает объект "Графическая метка"                            |
//+------------------------------------------------------------------+
bool BitmapLabelMove(const long   chart_ID=0,       // ID графика
                     const string name="BmpLabel", // имя метки
                     const int    x=0,             // координата по оси X
                     const int    y=0)             // координата по оси Y
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим объект
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": не удалось переместить X-координату объекта! Код ошибки = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": не удалось переместить Y-координату объекта! Код ошибки = ",GetLastErro
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет размер области видимости (размер объекта)               |
//+------------------------------------------------------------------+
bool BitmapLabelChangeSize(const long   chart_ID=0,       // ID графика
                           const string name="BmpLabel", // имя метки
                           const int    width=0,         // ширина метки
                           const int    height=0)        // высота метки
  {
//--- сбросим значение ошибки
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- изменим размеры объекта
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": не удалось изменить ширину объекта! Код ошибки = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": не удалось изменить высоту объекта! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет координату левого верхнего угла области видимости       |
//+------------------------------------------------------------------+
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,      // ID графика
                                const string name="BmpLabel", // имя метки
                                const int    x_offset=0,      // координата X области
                                const int    y_offset=0)      // координата Y области
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим координаты области видимости объекта
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": не удалось изменить X-координату области видимости! Код ошибки = ",GetL
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": не удалось изменить Y-координату области видимости! Код ошибки = ",GetL
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет объект "Графическая метка"                               |
//+------------------------------------------------------------------+
bool BitmapLabelDelete(const long   chart_ID=0,      // ID графика
                       const string name="BmpLabel") // имя метки
  {
//--- сбросим значение ошибки
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- удалим метку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить объект \"Графическая метка\"! Код ошибки = ",GetLast
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- размеры окна графика
   long x_distance;
   long y_distance;
//--- определим размеры окна
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Не удалось получить ширину графика! Код ошибки = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Не удалось получить высоту графика! Код ошибки = ",GetLastError());
      return;
     }
//--- определим координаты графической метки
   int x=(int)x_distance/2;
   int y=(int)y_distance/2;
//--- установим размеры метки и координаты области видимости
   int width=32;
   int height=32;
   int x_offset=0;
   int y_offset=0;
//--- разместим графическую метку по центру окна
   if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn,InpFileOff,width,height,x_offset,y_
      InpCorner,InpAnchor,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidd
     {
      return;
     }
//--- перерисуем график и подождем одну секунду
   ChartRedraw();
   Sleep(1000);
//--- изменим размеры области видимости метки в цикле
   for(int i=0;i<6;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- изменим размеры области видимости
       width--;
       height--;
       if(!BitmapLabelChangeSize(0,InpName,width,height))
          return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
          return;
       //--- перерисуем график
       ChartRedraw();
       // задержка в 0.3 секунды
       Sleep(300);
      }
//--- задержка в 1 секунду
   Sleep(1000);
//--- изменим координаты области видимости метки в цикле
   for(int i=0;i<2;i++)
     {
       //--- изменим координаты области видимости
       x_offset++;
       y_offset++;
       if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
          return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
          return;
       //--- перерисуем график
       ChartRedraw();
       // задержка в 0.3 секунды
       Sleep(300);
      }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим метку
   BitmapLabelDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EDIT

Объект "Поле ввода".



Примечание

Координаты точки привязки задаются в пикселях. Можно выбрать угол привязки "Поля ввода" из перечисления ENUM_BASE_CORNER.

Также можно выбрать один из типов выравнивания текста внутри "Поля ввода" из перечисления ENUM_ALIGN_MODE.

Пример

Следующий скрипт создает и перемещает на графике объект "Поле ввода". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает объект \"Поле ввода\"."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string            InpName="Edit";              // Имя объекта
input string            InpText="Text";              // Текст объекта
input string            InpFont="Arial";             // Шрифт
input int               InpFontSize=14;              // Размер шрифта
input ENUM_ALIGN_MODE   InpAlign=ALIGN_CENTER;       // Способ выравнивания текста
input bool              InpReadOnly=false;           // Возможность редактировать
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER; // Угол графика для привязки
input color             InpColor=clrBlack;           // Цвет текста
```

© 2000-2025, MetaQuotes Ltd.

```
input color             InpBackColor=clrWhite;       // Цвет фона
input color             InpBorderColor=clrBlack;     // Цвет границы
input bool              InpBack=false;               // Объект на заднем плане
input bool              InpSelection=false;          // Выделить для перемещений
input bool              InpHidden=true;              // Скрыт в списке объектов
input long              InpZOrder=0;                 // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает объект "Поле ввода"                                      |
//+------------------------------------------------------------------+
bool EditCreate(const long             chart_ID=0,               // ID графика
                const string           name="Edit",              // имя объекта
                const int              sub_window=0,             // номер подокна
                const int              x=0,                      // координата по оси
                const int              y=0,                      // координата по оси
                const int              width=50,                 // ширина
                const int              height=18,                // высота
                const string           text="Text",              // текст
                const string           font="Arial",             // шрифт
                const int              font_size=10,             // размер шрифта
                const ENUM_ALIGN_MODE  align=ALIGN_CENTER,       // способ выравнивани
                const bool             read_only=false,          // возможность редакт
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // угол графика для
                const color            clr=clrBlack,             // цвет текста
                const color            back_clr=clrWhite,        // цвет фона
                const color            border_clr=clrNONE,       // цвет границы
                const bool             back=false,               // на заднем плане
                const bool             selection=false,          // выделить для перем
                const bool             hidden=true,              // скрыт в списке об
                const long             z_order=0)                // приоритет на нажат
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим поле ввода
   if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать объект \"Поле ввода\"! Код ошибки = ",GetLastError()
      return(false);
     }
//--- установим координаты объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- установим размеры объекта
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- установим текст
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- установим шрифт текста
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим размер шрифта
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- установим способ выравнивания текста в объекте
   ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- установим (true) или отменим (false) режим только для чтения
   ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- установим угол графика, относительно которого будут определяться координаты объе
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- установим цвет текста
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим цвет фона
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- установим цвет границы
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения метки мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает объект "Поле ввода"                                   |
//+------------------------------------------------------------------+
bool EditMove(const long   chart_ID=0,  // ID графика
              const string name="Edit", // имя объекта
              const int    x=0,         // координата по оси X
              const int    y=0)         // координата по оси Y
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим объект
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": не удалось переместить X-координату объекта! Код ошибки = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": не удалось переместить Y-координату объекта! Код ошибки = ",GetLastErro
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет размеры объекта "Поле ввода"                            |
//+------------------------------------------------------------------+
bool EditChangeSize(const long   chart_ID=0,  // ID графика
                    const string name="Edit", // имя объекта
                    const int    width=0,     // ширина
                    const int    height=0)    // высота
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим размеры объекта
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": не удалось изменить ширину объекта! Код ошибки = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": не удалось изменить высоту объекта! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет текст объекта "Поле ввода"                              |
//+------------------------------------------------------------------+
bool EditTextChange(const long   chart_ID=0,  // ID графика
                    const string name="Edit", // имя объекта
                    const string text="Text") // текст
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим текст объекта
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": не удалось изменить текст! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Возвращает текст объекта "Поле ввода"                            |
//+------------------------------------------------------------------+
bool EditTextGet(string      &text,         // текст
                 const long   chart_ID=0,  // ID графика
                 const string name="Edit") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- получим текст объекта
   if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
     {
      Print(__FUNCTION__,
            ": не удалось получить текст! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет объект "Поле ввода"                                      |
//+------------------------------------------------------------------+
bool EditDelete(const long   chart_ID=0,  // ID графика
                const string name="Edit") // имя объекта
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим метку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": не удалось удалить объект \"Поле ввода\"! Код ошибки = ",GetLastError()
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- размеры окна графика
   long x_distance;
   long y_distance;
//--- определим размеры окна
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Не удалось получить ширину графика! Код ошибки = ",GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
      {
       Print("Не удалось получить высоту графика! Код ошибки = ",GetLastError());
       return;
      }
//--- определим шаг для изменения размеров поля ввода
   int x_step=(int)x_distance/64;
//--- установим координаты поля ввода и его размер
   int x=(int)x_distance/8;
   int y=(int)y_distance/2;
   int x_size=(int)x_distance/8;
   int y_size=InpFontSize*2;
//--- запомним текст в локальную переменную
   string text=InpText;
//--- создадим поле ввода
   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;
      }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- растянем поле ввода
   while(x_size-x<x_distance*5/8)
      {
       //--- увеличим ширину поля ввода
       x_size+=x_step;
       if(!EditChangeSize(0,InpName,x_size,y_size))
          return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
          return;
       //--- перерисуем график и подождем 0.05 секунды
       ChartRedraw();
       Sleep(50);
      }
//--- задержка в полсекунды
   Sleep(500);
//--- изменим текст
   for(int i=0;i<20;i++)
      {
       //--- прибавим "+" в начале и конце
       text="+"+text+"+";
       if(!EditTextChange(0,InpName,text))
          return;
       //--- проверим факт принудительного завершения скрипта
       if(IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
      //--- перерисуем график и подождем 0.1 секунды
      ChartRedraw();
      Sleep(100);
     }
//--- задержка в полсекунды
   Sleep(500);
//--- удалим поле ввода
   EditDelete(0,InpName);
   ChartRedraw();
//--- подождем 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EVENT

Объект "Событие".



Примечание

При наведении на событие курсора мыши, всплывает его текст.

Пример

Следующий скрипт создает и перемещает на графике объект "Событие". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт строит графический объект \"Событие\"."
#property description "Дата точки привязки задается в процентах от"
#property description "ширины окна графика в барах."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="Event";     // Имя события
input int              InpDate=25;          // Дата события в %
input string           InpText="Text";      // Текст события
input color            InpColor=clrRed;     // Цвет события
input int              InpWidth=1;          // Размер точки при выделении
input bool             InpBack=false;       // Событие на заднем плане
input bool             InpSelection=false; // Выделить для перемещений
input bool             InpHidden=true;      // Скрыт в списке объектов
```

© 2000-2025, MetaQuotes Ltd.

```
input long            InpZOrder=0;        // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает объект "Событие" на графике                              |
//+------------------------------------------------------------------+
bool EventCreate(const long            chart_ID=0,       // ID графика
                 const string          name="Event",     // имя события
                 const int             sub_window=0,     // номер подокна
                 const string          text="Text",      // текст события
                 datetime              time=0,           // время
                 const color           clr=clrRed,       // цвет
                 const int             width=1,          // толщина точки при выделении
                 const bool            back=false,       // на заднем плане
                 const bool            selection=false,  // выделить для перемещений
                 const bool            hidden=true,      // скрыт в списке объектов
                 const long            z_order=0)        // приоритет на нажатие мышью
  {
//--- если время не задано, то создаем объект на последнем баре
   if(!time)
      time=TimeCurrent();
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим объект "Событие"
   if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать объект \"Событие\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим текст события
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- установим цвет
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим толщину точки привязки, если объект выделен
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения события мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет текст объекта "Событие"                                 |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool EventTextChange(const long   chart_ID=0,   // ID графика
                     const string name="Event", // имя события
                     const string text="Text")  // текст
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим текст объекта
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": не удалось изменить текст! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещение объект "Событие"                                     |
//+------------------------------------------------------------------+
bool EventMove(const long   chart_ID=0,   // ID графика
               const string name="Event", // имя события
               datetime     time=0)       // время
  {
//--- если время не задано, то перемещаем событие на последний бар
   if(!time)
      time=TimeCurrent();
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим объект
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": не удалось переместить объект \"Событие\"! Код ошибки = ",GetLastError
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет объект "Событие"                                         |
//+------------------------------------------------------------------+
bool EventDelete(const long   chart_ID=0,   // ID графика
                 const string name="Event") // имя события
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим объект
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": не удалось удалить объект \"Событие\"! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим входные параметры на корректность
   if(InpDate<0 || InpDate>100)
     {
      Print("Ошибка! Некорректные значения входных параметров!");
      return;
     }
//--- количество видимых баров в окне графика
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- массив для хранения значений дат, которые будут использованы
//--- для установки и изменения координаты точки привязки объекта "Событие"
   datetime date[];
//--- выделение памяти
   ArrayResize(date,bars);
//--- заполним массив дат
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Не удалось скопировать значения времени! Код ошибки = ",GetLastError());
      return;
     }
//--- определим точки для создания объекта
   int d=InpDate*(bars-1)/100;
//--- создадим объект "Событие"
   if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- теперь будем перемещать объект
//--- счетчик цикла
   int h_steps=bars/2;
//--- перемещаем объект
   for(int i=0;i<h_steps;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- возьмем следующее значение
      if(d<bars-1)
         d+=1;
      //--- сдвигаем точку
      if(!EventMove(0,InpName,date[d]))
         return;
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график
      ChartRedraw();
      // задержка в 0.05 секунды
      Sleep(50);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- удалим канал с графика
   EventDelete(0,InpName);
   ChartRedraw();
//--- задержка в 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_RECTANGLE_LABEL

Объект "Прямоугольная метка".



Примечание

Координаты точки привязки задаются в пикселях. Можно выбрать угол привязки прямоугольной метки из перечисления ENUM_BASE_CORNER. Тип рамки для прямоугольной метки можно выбрать из перечисления ENUM_BORDER_TYPE.

Предназначена для создания и оформления пользовательского графического интерфейса.

Пример

Следующий скрипт создает и перемещает на графике объект "Прямоугольная метка". Для создания и изменения свойств графического объекта написаны специальные функции, которые вы можете использовать "как есть" в своих собственных программах.

```
//--- описание
#property description "Скрипт создает графический объект \"Прямоугольная метка\"."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры скрипта
input string           InpName="RectLabel";          // Имя метки
input color            InpBackColor=clrSkyBlue;      // Цвет фона
input ENUM_BORDER_TYPE InpBorder=BORDER_FLAT;        // Тип рамки
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Угол графика для привязки
input color            InpColor=clrDarkBlue;         // Цвет плоской границы (Flat)
input ENUM_LINE_STYLE  InpStyle=STYLE_SOLID;         // Стиль плоской границы (Flat)
input int              InpLineWidth=3;               // Толщина плоской границы (Flat)
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpBack=false;               // Объект на заднем плане
input bool              InpSelection=true;           // Выделить для перемещений
input bool              InpHidden=true;              // Скрыт в списке объектов
input long              InpZOrder=0;                 // Приоритет на нажатие мышью
//+------------------------------------------------------------------+
//| Создает прямоугольную метку                                      |
//+------------------------------------------------------------------+
bool RectLabelCreate(const long             chart_ID=0,               // ID графика
                     const string           name="RectLabel",         // имя метки
                     const int              sub_window=0,             // номер подокна
                     const int              x=0,                      // координата по
                     const int              y=0,                      // координата по
                     const int              width=50,                 // ширина
                     const int              height=18,                // высота
                     const color            back_clr=C'236,233,216',  // цвет фона
                     const ENUM_BORDER_TYPE border=BORDER_SUNKEN,     // тип границы
                     const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // угол графика
                     const color            clr=clrRed,               // цвет плоской
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,        // стиль плоской
                     const int              line_width=1,             // толщина плоск
                     const bool             back=false,               // на заднем пла
                     const bool             selection=false,          // выделить для
                     const bool             hidden=true,              // скрыт в списк
                     const long             z_order=0)                // приоритет на
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- создадим прямоугольную метку
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": не удалось создать прямоугольную метку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- установим координаты метки
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- установим размеры метки
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- установим цвет фона
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- установим тип границы
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//--- установим угол графика, относительно которого будут определяться координаты точк
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- установим цвет плоской рамки (в режиме Flat)
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- установим стиль линии плоской рамки
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- установим толщину плоской границы
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- отобразим на переднем (false) или заднем (true) плане
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- включим (true) или отключим (false) режим перемещения метки мышью
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- скроем (true) или отобразим (false) имя графического объекта в списке объектов
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- установим приоритет на получение события нажатия мыши на графике
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Перемещает прямоугольную метку                                   |
//+------------------------------------------------------------------+
bool RectLabelMove(const long   chart_ID=0,       // ID графика
                   const string name="RectLabel", // имя метки
                   const int    x=0,              // координата по оси X
                   const int    y=0)              // координата по оси Y
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- переместим прямоугольную метку
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": не удалось переместить X-координату метки! Код ошибки = ",GetLastError
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": не удалось переместить Y-координату метки! Код ошибки = ",GetLastError
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет размер прямоугольной метки                              |
//+------------------------------------------------------------------+
bool RectLabelChangeSize(const long   chart_ID=0,       // ID графика
                         const string name="RectLabel", // имя метки
                         const int    width=50,         // ширина метки
                         const int    height=18)        // высота метки
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим размеры метки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": не удалось изменить ширину метки! Код ошибки = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": не удалось изменить высоту метки! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Изменяет тип границы прямоугольной метки                         |
//+------------------------------------------------------------------+
bool RectLabelChangeBorderType(const long             chart_ID=0,           // ID граф
                               const string           name="RectLabel",     // имя мет
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // тип гра
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- изменим тип рамки
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
     {
      Print(__FUNCTION__,
            ": не удалось изменить тип границы! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет прямоугольную метку                                      |
//+------------------------------------------------------------------+
bool RectLabelDelete(const long   chart_ID=0,       // ID графика
                     const string name="RectLabel") // имя метки
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- удалим метку
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": не удалось удалить прямоугольную метку! Код ошибки = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- размеры окна графика
   long x_distance;
   long y_distance;
//--- определим размеры окна
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Не удалось получить ширину графика! Код ошибки = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Не удалось получить высоту графика! Код ошибки = ",GetLastError());
      return;
     }
//--- определим координаты прямоугольной метки
   int x=(int)x_distance/4;
   int y=(int)y_distance/4;
//--- установим размеры метки
   int width=(int)x_distance/4;
   int height=(int)y_distance/4;
//--- создадим прямоугольную метку
   if(!RectLabelCreate(0,InpName,0,x,y,width,height,InpBackColor,InpBorder,InpCorner,
      InpColor,InpStyle,InpLineWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- перерисуем график и подождем одну секунду
   ChartRedraw();
   Sleep(1000);
//--- изменим размер прямоугольной метки
   int steps=(int)MathMin(x_distance/4,y_distance/4);
   for(int i=0;i<steps;i++)
     {
      //--- изменим размеры
      width+=1;
      height+=1;
      if(!RectLabelChangeSize(0,InpName,width,height))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- проверим факт принудительного завершения скрипта
      if(IsStopped())
         return;
      //--- перерисуем график и подождем 0.01 секунды
      ChartRedraw();
      Sleep(10);
     }
//--- задержка в 1 секунду
   Sleep(1000);
//--- изменим тип рамки
   if(!RectLabelChangeBorderType(0,InpName,BORDER_RAISED))
      return;
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- изменим тип рамки
   if(!RectLabelChangeBorderType(0,InpName,BORDER_SUNKEN))
      return;
//--- перерисуем график и подождем 1 секунду
   ChartRedraw();
   Sleep(1000);
//--- удалим метку
   RectLabelDelete(0,InpName);
   ChartRedraw();
//--- подождем 1 секунду
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

Свойства объектов

Графические объекты могут иметь множество свойств в зависимости от типа объекта. Установка и получение значений свойств объектов производится соответствующими функциями по работе с графическими объектами.

Все объекты, используемые в техническом анализе, имеют привязку на графиках по координатам цены и времени - трендовая линия, каналы, инструменты Фибоначчи и т.д. Но есть ряд вспомогательных объектов, предназначенных для улучшения интерфейса, которые имеют привязку к видимой всегда части графика (основное окно графика или подокна индикаторов):

Объект	Идентификатор	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	–	–	Да	–	Да	Да
Label	OBJ_LABEL	Да	Да (только для чтения)	–	Да	Да	Да
Button	OBJ_BUTTON	Да	Да	–	Да	–	–
Bitmap	OBJ_BITMAP	–	Да (только для чтения)	Да	–	Да	–
Bitmap Label	OBJ_BITMAP_LABEL	Да	Да (только для чтения)	–	Да	Да	–
Edit	OBJ_EDIT	Да	Да	–	Да	–	–
Rectangle Label	OBJ_RECTANGLE_LABEL	Да	Да	–	Да	–	–

В таблице использованы следующие обозначения:

- **X/Y** - координаты точки привязки задаются в пикселях относительно одного из углов графика;
- **Width/Height** - объекты имеет ширину и высоту. Если указано "только для чтения", то это означает, что значения ширины и высоты вычисляются только после отрисовки объекта на графике;
- **Date/Price** - координаты точки привязки задаются парой дата/цена;
- **OBJPROP_CORNER** - задаёт угол графика, относительно которого указываются координаты точки привязки. Может быть одним из 4-х значений перечисления ENUM_BASE_CORNER;

© 2000-2025, MetaQuotes Ltd.

- **OBJPROP_ANCHOR** - задаёт положение точки привязки в самом объекте, и может быть одним из 9-ти значений перечисления ENUM_ANCHOR_POINT. Именно от этой точки до выбранного угла графика указываются координаты в пикселях;
- **OBJPROP_ANGLE** - задаёт угол поворота объекта против часовой стрелки.

Функции, задающие свойства графических объектов, а также операции создания ObjectCreate() и перемещения ObjectMove() объектов на графике фактически служат для отправки команд графику. При успешном выполнении этих функций команда попадает в общую очередь событий графика. Визуальное изменение свойств графических объектов производится в процессе обработки очереди событий данного графика.

По этой причине не следует ожидать немедленного визуального обновления графических объектов после вызова данных функций. В общем случае обновление графических объектов на чарте производится терминалом автоматически по событиям изменения - поступление новой котировки, изменения размера окна графика и т.д. Для принудительного обновления графических объектов используйте команду на перерисовку графика ChartRedraw().

Для функций ObjectSetInteger() и ObjectGetInteger()

ENUM_OBJECT_PROPERTY_INTEGER

Идентификатор	Описание	Тип свойства
OBJPROP_COLOR	Цвет	color
OBJPROP_STYLE	Стиль	ENUM_LINE_STYLE
OBJPROP_WIDTH	Толщина линии	int
OBJPROP_BACK	Объект на заднем плане	bool
OBJPROP_ZORDER	Приоритет графического ког	long

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	объекта на получение события нажатия мышки на графике (CHARTEVENT_CLICK). По умолчанию при создании значение выставляется равны	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	мнулю, но при необходимости можно повысить приоритет. При наложении объектов друг на друга событие CHARTEVENT_CLICK получит тол	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ько один объект, чей приоритет выше остальных.	
OBJPROP_FILL	Заливка объекта цветом (для OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	_STDDEVCHANNEL, OBJ_REGRESSION)	
OBJPROP_HIDDEN	Запрет на показ имени графического объекта в списке объектов изменю терминала "Графики" - "Объекты" -	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	"Список объектов". Значение true позволяет скрыть ненужный для пользователя объект из списка. По умолчанию true устанавливается для объект	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ов, которые отображают события календаря, историю торговли, а также для создания из MQL5-программы. Для того чтобы увидеть такие	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	графические объекты и получить доступ к их свойствам, нужно нажать кнопку "Все" в окне "Список объектов".	
OBJPROP_SELECTED	Выделенность объекта	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
OBJPROP_READONLY	Возможность редактирования текста в объекте Edit	bool
OBJPROP_TYPE	Тип объекта	ENUM_OBJECT r/o
OBJPROP_TIME	Координата времени	datetime модификатор=номер точки привязки
OBJPROP_SELECTABLE	Доступность объекта	bool
OBJPROP_CREATETIME	Время создания объекта	datetime r/o

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
OBJPROP_LEVELS	Количество уровней	int
OBJPROP_LEVELCOLOR	Цвет линии-уровня	color модификатор=номер уровня
OBJPROP_LEVELSTYLE	Стиль линии-уровня	ENUM_LINE_STYLE модификатор=номер уровня
OBJPROP_LEVELWIDTH	Толщина линии-уровня	int модификатор=номер уровня
OBJPROP_ALIGN	Горизонтальное выравнивание текста в объекте "Поле вво	ENUM_ALIGN_MODE

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	да" (OBJ_EDIT)	
OBJPROP_FONTSIZE	Размер шрифта	int
OBJPROP_RAY_LEFT	Луч продолжается влево	bool
OBJPROP_RAY_RIGHT	Луч продолжается вправо	bool
OBJPROP_RAY	Вертикальная линия продолжается на все окна графика	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
OBJPROP_ELLIPSE	Отображение полного эллипса для объекта "Дуги Фибоначчи" (OBJ_FIBOARC)	bool
OBJPROP_ARROWCODE	Код стрелки для объекта "Стрелка"	uchar
OBJPROP_TIMEFRAMES	Видимость объекта на тай	набор флагов flags

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	мфреймах	
OBJPROP_ANCHOR	Положение точки привязки графического объекта	ENUM_ARROW_ANCHOR (для OBJ_ARROW), ENUM_ANCHOR_POINT (для OBJ_LABEL, OBJ_BITMAP_LABEL и OBJ_TEXT)
OBJPROP_XDISTANCE	Дистанция в пикселях по оси X от угла привязки (см. примечание)	int
OBJPROP_YDISTANCE	Дистан	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ция в пикселях по оси Y от угла привязки (см. примечание)	
OBJPROP_DIRECTION	Тренд объекта Ганна	ENUM_GANN_DIRECTION
OBJPROP_DEGREE	Уровень волновой разметки Эллиота	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	Отображение линий	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	для волновой разметки Эллиота	
OBJPROP_STATE	Состояние кнопки (Нажата/Отжата)	bool
OBJPROP_CHART_ID	Идентификатор объекта "График" (OBJ_CHART). Позволяет работать со свойст	long r/o

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ва ми это го объ ект а как с об ыч ны м гра фи ко м с по мо щь ю фу нкц ий из раз дел а Оп ера ции с гра фи ка ми, но ест ь нек ото ры е иск лю чен ия.	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
OBJPROP_XSIZE	Ширина объекта по оси X в пикселях. Задается для объектов OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT,	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	OBJ_RECTANGLE_LABEL.	
OBJPROP_YSIZE	Высота объекта по оси Y в пикселях. Задается для объектов OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BI	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	TMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_XOFFSET	X-координата левого верхнего угла прямоугольной области видимости в графических объектах "Гр	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	графическая метка" и "Рисунок" (OBJ_BITMAP_LABEL и OBJ_BITMAP). Значение задается в пикселях относительного верхнего левого угла исходного	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	изображения.	
OBJPROP_YOFFSET	Y-координата левого верхнего угла прямоугольной области видимости в графических объектах "Графическая метка" и "Рисунок"	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	(OBJ_BITMAP_LABEL и OBJ_BITMAP). Значение задается в пикселях относительного верхнего левого угла исходного изображения.	
OBJPROP_PERIOD	Период для объект	ENUM_TIMEFRAMES

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	а "График"	
OBJPROP_DATE_SCALE	Признак отображения шкалы времени для объекта "График"	bool
OBJPROP_PRICE_SCALE	Признак отображения ценовой шкалы для объекта "График"	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
OBJPROP_CHART_SCALE	Масштаб для объекта "График"	int значение в диапазоне 0-5
OBJPROP_BGCOLOR	Цвет фона для OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	color
OBJPROP_CORNER	Угол графика для привязки графического об	ENUM_BASE_CORNER

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	екта	
OBJPROP_BORDER_TYPE	Тип рамки для объекта "Прямоугольная рамка"	ENUM_BORDER_TYPE
OBJPROP_BORDER_COLOR	Цвет рамки для объекта OBJ_EDIT и OBJ_BUTTON	color

При применении операций с графиками для объекта "График" (OBJ_CHART) действуют следующие ограничения:

- нельзя закрыть с помощью ChartClose();
- нельзя поменять символ/период с помощью функции ChartSetSymbolPeriod();
- не работают свойства CHART_SCALE, CHART_BRING_TO_TOP, CHART_SHOW_DATE_SCALE и CHART_SHOW_PRICE_SCALE (ENUM_CHART_PROPERTY_INTEGER).

© 2000-2025, MetaQuotes Ltd.

Для объектов OBJ_BITMAP_LABEL и OBJ_BITMAP программным путем можно установить специальный режим показа изображения. В этом режиме показывается только та часть исходного изображения, на которую накладывается прямоугольная область видимости, остальная часть картинки становится невидимой. Размеры области видимости необходимо установить с помощью свойств OBJPROP_XSIZE и OBJPROP_YSIZE. Область видимости можно "перемещать" только в пределах исходного изображения с помощью свойств OBJPROP_XOFFSET и OBJPROP_YOFFSET.

Для объектов с фиксированными размерами: OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT и OBJ_CHART свойства OBJPROP_XDISTANCE и OBJPROP_YDISTANCE задают положение левой верхней точки объекта относительно угла графика (OBJPROP_CORNER), от которого будут отсчитываться координаты X и Y в пикселях.

Для функций ObjectSetDouble() и ObjectGetDouble()

ENUM_OBJECT_PROPERTY_DOUBLE

Идентификатор	Описание	Тип свойства
OBJPROP_PRICE	Координата цены	double модификатор=номер точки привязки
OBJPROP_LEVELVALUE	Значение уровня	double модификатор=номер уровня
OBJPROP_SCALE	Масштаб (свойство объектов Ганна и объекта "Ду	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ги Фибоначчи")	
OBJPROP_ANGLE	Угол. Для объектов свеще не заданным углом, созданных из программы, значение равно EMPTY_VALUE	double
OBJPROP_DEVIATION	Отклонение для канала	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	стандартного отклонения	

Для функций ObjectSetString() и ObjectGetString()

ENUM_OBJECT_PROPERTY_STRING

Идентификатор	Описание	Тип свойства
OBJPROP_NAME	Имя объекта	string
OBJPROP_TEXT	Описание объекта (текст, содержащийся в объекте)	string
OBJPROP_TOOLTIP	Текст всплыва	string

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ющей подсказки. Если свойство не задано, то показывается подсказка, автоматически сформированная терминалом. Можно отключить показ	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	подсказки, установив для нее значение "\n" (перевод строки)	
OBJPROP_LEVELTEXT	Описание уровня	string модификатор=номер уровня
OBJPROP_FONT	Шрифт	string
OBJPROP_BMPFILE	Имя BMP-файла для объекта "Графическая метка".	string модификатор: 0-состояние ON, 1-состояние OFF

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	Смотри также Ресурсы	
OBJPROP_SYMBOL	Символ для объекта "График"	string

Для объекта OBJ_RECTANGLE_LABEL ("Прямоугольная метка") можно задать один из трех видов отображения, которым соответствуют значения из перечисления ENUM_BORDER_TYPE.

ENUM_BORDER_TYPE

Идентификатор	Описание
BORDER_FLAT	Плоский вид
BORDER_RAISED	Выпуклый вид
BORDER_SUNKEN	Вогнутый вид

Для объекта OBJ_EDIT ("Поле ввода") и для функции ChartScreenShot() можно указать тип выравнивания по горизонтали с помощью значений перечисления ENUM_ALIGN_MODE.

ENUM_ALIGN_MODE

Идентификатор	Описание
ALIGN_LEFT	Выравнивание по левой границе
ALIGN_CENTER	Выравнивание по центру (только для объекта "Поле ввода")
ALIGN_RIGHT	Выравнивание по правой границе

© 2000-2025, MetaQuotes Ltd.

Пример:

```
#define  UP          "\x0431"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="my_OBJ_LABEL_object";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object ",label_name," not found. Error code = ",GetLastError());
      //--- создадим объект Label
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- установим координату X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- установим координату Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      //--- зададим цвет текста
      ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
      //--- установим текст для объекта Label
      ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
      //--- установим шрифт надписи
      ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
      //--- установим размер шрифта
      ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
      //--- повернем на 45 градусов по часовой стрелке
      ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
      //--- запретим выделение объекта мышкой
      ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
      //--- отрисуем на графике
      ChartRedraw(0);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Способы привязки объектов

Графические объекты Text, Label, Bitmap и Bitmap Label (OBJ_TEXT, OBJ_LABEL, OBJ_BITMAP и OBJ_BITMAP_LABEL) могут иметь один из 9 различных способов привязки своих координат, задаваемых свойством OBJPROP_ANCHOR.

Объект	Идентификатор	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	–	–	Да	–	**Да**	Да
Label	OBJ_LABEL	Да	Да (только для чтения)	–	Да	**Да**	Да
Button	OBJ_BUTTON	Да	Да	–	Да	–	–
Bitmap	OBJ_BITMAP	–	Да (только для чтения)	Да	–	**Да**	–
Bitmap Label	OBJ_BITMAP_LABEL	Да	Да (только для чтения)	–	Да	**Да**	–
Edit	OBJ_EDIT	Да	Да	–	Да	–	–
Rectangle Label	OBJ_RECTANGLE_LABEL	Да	Да	–	Да	–	–

В таблице использованы следующие обозначения:

- **X/Y** - координаты точки привязки задаются в пикселях относительно одного из углов графика;
- **Width/Height** - объекты имеет ширину и высоту. Если указано "только для чтения", то это означает, что значения ширины и высоты вычисляются только после отрисовки объекта на графике;
- **Date/Price** - координаты точки привязки задаются парой дата/цена;
- **OBJPROP_CORNER** - задаёт угол графика, относительно которого указываются координаты точки привязки. Может быть одним из 4-х значений перечисления ENUM_BASE_CORNER;
- **OBJPROP_ANCHOR** - задаёт положение точки привязки в самом объекте, и может быть одним из 9-ти значений перечисления ENUM_ANCHOR_POINT. Именно от этой точки до выбранного угла графика указываются координаты в пикселях;
- **OBJPROP_ANGLE** - задаёт угол поворота объекта против часовой стрелки.

© 2000-2025, MetaQuotes Ltd.

Указать нужный вариант можно с помощью функции ObjectSetInteger(handle_графика, имя_объекта, OBJPROP_ANCHOR, способ_привязки), где способ_привязки - одно из значений перечисления ENUM_ANCHOR_POINT.

ENUM_ANCHOR_POINT

Идентификатор	Описание
ANCHOR_LEFT_UPPER	Точка привязки в левом верхнем углу
ANCHOR_LEFT	Точка привязки слева по центру
ANCHOR_LEFT_LOWER	Точка привязки в левом нижнем углу
ANCHOR_LOWER	Точка привязки снизу по центру
ANCHOR_RIGHT_LOWER	Точка привязки в правом нижнем углу
ANCHOR_RIGHT	Точка привязки справа по центру
ANCHOR_RIGHT_UPPER	Точка привязки в правом верхнем углу
ANCHOR_UPPER	Точка привязки сверху по центру
ANCHOR_CENTER	Точка привязки строго по центру объекта

Объекты OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT и OBJ_CHART имеют фиксированную точку привязки в левом верхнем углу (ANCHOR_LEFT_UPPER).

Пример:

```
string text_name="my_OBJ_TEXT_object";
if(ObjectFind(0,text_name)<0)
  {
   Print("Object ",text_name," not found. Error code = ",GetLastError());
   //--- получим максимальную цену графика
   double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
   //--- создадим объект Label
   ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
   //--- зададим цвет текста
   ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
   //--- зададим цвет фона
   ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
   //--- установим текст для объекта Label
   ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
   //--- установим шрифт надписи
   ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
   //--- установим размер шрифта
   ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
   //--- установим привязку к правому верхнему углу
   ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
   //--- повернем на 90 градусов против часовой стрелке
   ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
   //--- запретим выделение объекта мышкой
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
      //--- отрисуем на графике
      ChartRedraw(0);
     }
```

Графические объекты Arrow (OBJ_ARROW) имеют только 2 способа привязки своих координат. Идентификаторы перечислены в ENUM_ARROW_ANCHOR.

ENUM_ARROW_ANCHOR

Идентификатор	Описание
ANCHOR_TOP	Точка привязки для стрелки находится сверху
ANCHOR_BOTTOM	Точка привязки для стрелки находится внизу

Пример:

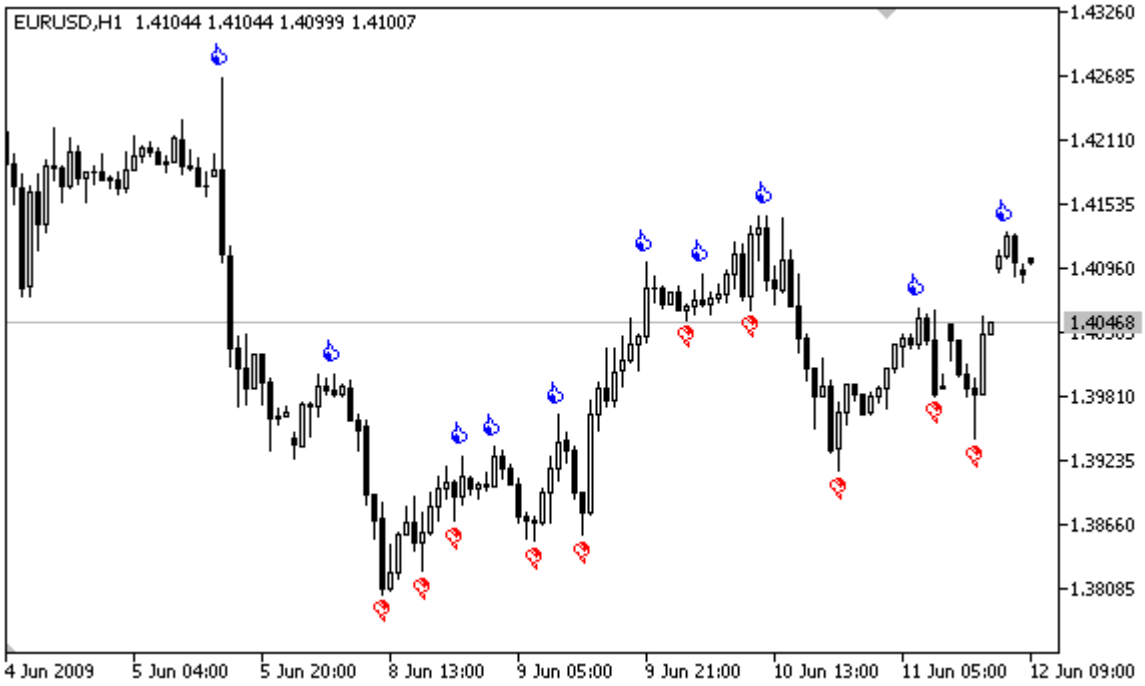
```
void OnStart()
  {
//--- служебные массивы
   double Ups[],Downs[];
   datetime Time[];
//--- установим для массивов признак таймсерии
   ArraySetAsSeries(Ups,true);
   ArraySetAsSeries(Downs,true);
   ArraySetAsSeries(Time,true);
//--- создадим хэндл индикатора Fractals
   int FractalsHandle=iFractals(NULL,0);
   Print("FractalsHandle = ",FractalsHandle);
//--- сбросим код ошибки
   ResetLastError();
//--- попытаемся скопировать значения индикатора
   int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Print("Не удалось скопировать верхние фракталы. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- попытаемся скопировать значения индикатора
   copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
   if(copied<=0)
     {
      Print("Не удалось скопировать нижние фракталы. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- скопируем таймсерию,содержащую время открытия последних 1000 баров
   copied=CopyTime(NULL,0,0,1000,Time);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(copied<=0)
     {
      Print("Не удалось скопировать времена открытия за последние 1000 баров");
      return;
     }

   int upcounter=0,downcounter=0; // будем в них подсчитывать количество стрелок
   bool created;// будем получать результат попытки создания объекта
   for(int i=2;i<copied;i++)// пробежимся по значениям индикатора iFractals
     {
      if(Ups[i]!=EMPTY_VALUE)// нашли верхний фрактал
        {
         if(upcounter<10)// создаем не более 10 объектов "вверх"
           {
            //--- попробуем создать объект "вверх"
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]
            if(created)// если создался - сделаем ему тюнинг
              {
               //--- точка привязки снизу, чтобы не наезжать на бар
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
               //--- последний штрих - покрасим
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
               upcounter++;
              }
           }
        }
      if(Downs[i]!=EMPTY_VALUE)// нашли нижний фрактал
        {
         if(downcounter<10)// создаем не более 10 объектов "вниз"
           {
            //--- попробуем создать объект "вниз"
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Down
            if(created)// если создался - сделаем ему тюнинг
              {
               //--- точка привязки сверху, чтобы не наезжать на бар
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
               //--- последний штрих - покрасим
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
               downcounter++;
              }
           }
        }
     }
  }
```

После выполнения скрипта график будет выглядеть примерно как на этом рисунке.

© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

Угол графика, к которому привязан объект

Существует ряд графических объектов, для которых можно задавать угол графика, относительно которого указываются координаты в пикселях. Это следующие типы объектов (в скобках указаны идентификаторы типа объекта):

- Label (OBJ_LABEL); Текстовая метка
- Button (OBJ_BUTTON); Кнопка
- Bitmap Label (OBJ_BITMAP_LABEL); Графическая метка
- Edit (OBJ_EDIT); Поле ввода
- Rectangle Label (OBJ_RECTANGLE_LABEL). Прямоугольная метка

Объект	Идентификатор	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	–	–	Да	–	Да	Да
Label	OBJ_LABEL	Да	Да (только для чтения)	–	**Да**	Да	Да
Button	OBJ_BUTTON	Да	Да	–	**Да**	–	–
Bitmap	OBJ_BITMAP	–	Да (только для чтения)	Да	–	Да	–
Bitmap Label	OBJ_BITMAP_LABEL	Да	Да (только для чтения)	–	**Да**	Да	–
Edit	OBJ_EDIT	Да	Да	–	**Да**	–	–
Rectangle Label	OBJ_RECTANGLE_LABEL	Да	Да	–	**Да**	–	–

В таблице использованы следующие обозначения:

- **X/Y** - координаты точки привязки задаются в пикселях относительно одного из углов графика;
- **Width/Height** - объекты имеет ширину и высоту. Если указано "только для чтения", то это означает, что значения ширины и высоты вычисляются только после отрисовки объекта на графике;
- **Date/Price** - координаты точки привязки задаются парой дата/цена;
- **OBJPROP_CORNER** - задаёт угол графика, относительно которого указываются координаты точки привязки. Может быть одним из 4-х значений перечисления ENUM_BASE_CORNER;

© 2000-2025, MetaQuotes Ltd.

- **OBJPROP_ANCHOR** - задаёт положение точки привязки в самом объекте, и может быть одним из 9-ти значений перечисления ENUM_ANCHOR_POINT. Именно от этой точки до выбранного угла графика указываются координаты в пикселях;
- **OBJPROP_ANGLE** - задаёт угол поворота объекта против часовой стрелки.

Для того чтобы указать угол графика, от которого будут отсчитываться координаты X и Y в пикселях, необходимо воспользоваться функцией ObjectSetInteger(chartID, name, OBJPROP_CORNER, **chart_corner**), где:

- chartID - идентификатор графика;
- name - имя графического объекта;
- OBJPROP_CORNER - идентификатор свойства для задания угла привязки;
- **chart_corner** - требуемый угол графика, может принимать одно из значений перечисления ENUM_BASE_CORNER.

ENUM_BASE_CORNER

Идентификатор	Описание
CORNER_LEFT_UPPER	Центр координат в левом верхнем углу графика
CORNER_LEFT_LOWER	Центр координат в левом нижнем углу графика
CORNER_RIGHT_LOWER	Центр координат в правом нижнем углу графика
CORNER_RIGHT_UPPER	Центр координат в правом верхнем углу графика

Пример:

```
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
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Видимость объектов

Комбинация флагов видимости объекта определяет таймфреймы графика, на которых объект отображаем. Для установки/получения значения свойства OBJPROP_TIMEFRAMES можно использовать функции ObjectSetInteger()/ObjectGetInteger().

Константа	Значение	Описание
OBJ_NO_PERIODS	0	Объект не показывается ни на одном таймфрейме
OBJ_PERIOD_M1	0x00000001	Объект рисуется на 1-минутных графиках
OBJ_PERIOD_M2	0x00000002	Объект рисуется на 2-минутных графиках
OBJ_PERIOD_M3	0x00000004	Объект рисуется на 3-минутных графиках
OBJ_PERIOD_M4	0x00000008	Объект рисуется на 4-минутных графиках
OBJ_PERIOD_M5	0x00000010	Объект рисуется на 5-минутных графиках
OBJ_PERIOD_M6	0x00000020	Объект рисуется на 6-минутных графиках
OBJ_PERIOD_M10	0x00000040	Объект рисуется на 10-минутных графиках
OBJ_PERIOD_M12	0x00000080	Объект рисуется на 12-минутных графиках
OBJ_PERIOD_M15	0x00000100	Объект рисуется на 15-минутных графиках
OBJ_PERIOD_M20	0x00000200	Объект рисуется на 20-минутных графиках
OBJ_PERIOD_M30	0x00000400	Объект рисуется на 30-минутных графиках
OBJ_PERIOD_H1	0x00000800	Объект рисуется на 1-часовых графиках
OBJ_PERIOD_H2	0x00001000	Объект рисуется на 2-часовых графиках
OBJ_PERIOD_H3	0x00002000	Объект рисуется на 3-часовых графиках
OBJ_PERIOD_H4	0x00004000	Объект рисуется на 4-часовых графиках

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
OBJ_PERIOD_H6	0x00008000	Объект рисуется на 6-часовых графиках
OBJ_PERIOD_H8	0x00010000	Объект рисуется на 8-часовых графиках
OBJ_PERIOD_H12	0x00020000	Объект рисуется на 12-часовых графиках
OBJ_PERIOD_D1	0x00040000	Объект рисуется на дневных графиках
OBJ_PERIOD_W1	0x00080000	Объект рисуется на недельных графиках
OBJ_PERIOD_MN1	0x00100000	Объект рисуется на месячных графиках
OBJ_ALL_PERIODS	0x001fffff	Объект рисуется на всех таймфреймах

Флаги видимости можно комбинировать с помощью символа "|", например, комбинация флагов OBJ_PERIOD_M10|OBJ_PERIOD_H4 означает, что объект будет видим на 10-минутном и 4-часовом таймфреймах.

Пример:

```
void OnStart()
  {
//---
   string highlevel="PreviousDayHigh";
   string lowlevel="PreviousDayLow";
   double prevHigh;           // High предыдущего дня
   double prevLow;            // Low предыдущего дня
   double highs[],lows[];     // массивы для получения High и Low

//--- сбросим значение последней ошибки
   ResetLastError();
//--- получим 2 последних значения High на дневном таймфрейме
   int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
   if(highsgot>0) // если копирование прошло успешно
     {
      Print("Цены High за последние 2 дня получены успешно");
      prevHigh=highs[0]; // High предыдущего дня
      Print("prevHigh = ",prevHigh);
      if(ObjectFind(0,highlevel)<0) // объект с именем highlevel не найден
         ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // создадим объект гор. линия
      //--- зададим ценовой уровень для линии highlevel
      ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
      //--- установим видимость только для  PERIOD_M10 и PERIOD_H4
      ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   else
      Print("Не удалось получить цены High за последние 2 дня, Error = ",GetLastError
//--- сбросим значение последней ошибки
   ResetLastError();
//--- получим 2 последних значения Low на дневном таймфрейме
   int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
   if(lowsgot>0) // если копирование прошло успешно
     {
      Print("Цены Low за последние 2 дня получены успешно");
      prevLow=lows[0]; // Low предыдущего дня
      Print("prevLow = ",prevLow);
      if(ObjectFind(0,lowlevel)<0) // объект с именем lowlevel не найден
         ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // создадим объект гор. линия
      //--- зададим ценовой уровень для линии lowlevel
      ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
      //--- установим видимость только для  PERIOD_M10 и PERIOD_H4
      ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else
      Print("Не удалось получить цены Low за последние 2 дня, Error = ",GetLastError()

   ChartRedraw(0); // перерисуем график принудительно
  }
```

Смотри также

PeriodSeconds, Period, Периоды графиков, Дата и время

© 2000-2025, MetaQuotes Ltd.

Уровни волн Эллиотта

Волны Эллиотта представлены двумя графическими объектами типов OBJ_ELLIOTWAVE5 и OBJ_ELLIOTWAVE3. Для задания размера волны (способа маркировки волн) используется свойство OBJPROP_DEGREE, которому можно установить одно из значений перечисления ENUM_ELLIOT_WAVE_DEGREE.

ENUM_ELLIOT_WAVE_DEGREE

Константа	Описание
ELLIOTT_GRAND_SUPERCYCLE	Главный Суперцикл (Grand Supercycle)
ELLIOTT_SUPERCYCLE	Суперцикл (Supercycle)
ELLIOTT_CYCLE	Цикл (Cycle)
ELLIOTT_PRIMARY	Первичный цикл (Primary)
ELLIOTT_INTERMEDIATE	Промежуточное звено (Intermediate)
ELLIOTT_MINOR	Второстепенный цикл (Minor)
ELLIOTT_MINUTE	Минута (Minute)
ELLIOTT_MINUETTE	Секунда (Minuette)
ELLIOTT_SUBMINUETTE	Субсекунда (Subminuette)

Пример:

```
   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)))
        {
         //--- установим уровень разметки в INTERMEDIATE
         ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
         //--- включим показ линий между вершинами волн
         ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
         //--- установим цвет линий
         ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
         //--- установим толщину линий
         ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
         //--- зададим описание
         ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

Объекты Ганна

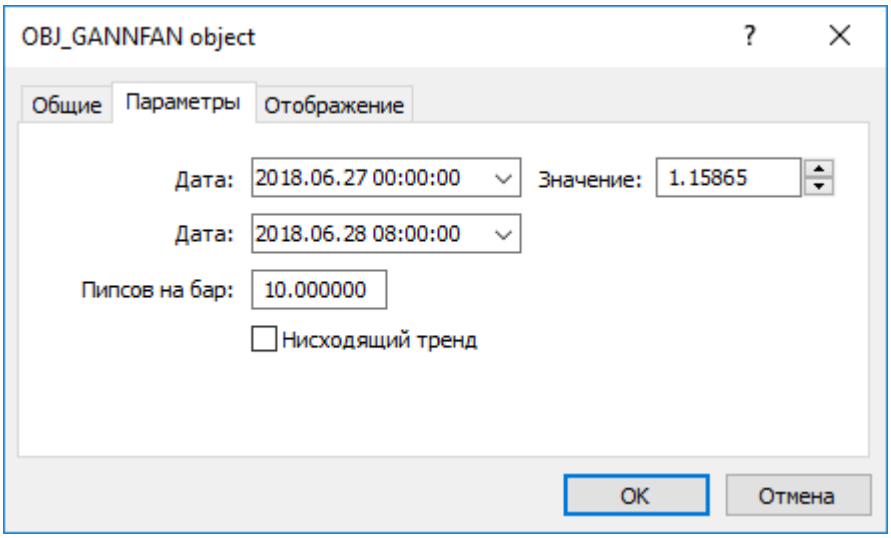
Для объектов веер Ганна (OBJ_GANNFAN) и сетка Ганна (OBJ_GANNGRID) можно указать одно из двух значений перечисления ENUM_GANN_DIRECTION, которое задает направление тренда.

ENUM_GANN_DIRECTION

Константа	Описание
GANN_UP_TREND	Линия соответствует восходящему тренду
GANN_DOWN_TREND	Линия соответствует нисходящему тренду

Для установки масштаба основной линии 1x1 используется функция ObjectSetDouble(chart_handle, gann_object_name, OBJPROP_SCALE, scale), где:

- chart_handle - окно графика, в котором находится объект;
- gann_object_name - имя объекта;
- OBJPROP_SCALE - идентификатор свойства "Scale";
- scale - требуемый масштаб в единицах Pips/Bar.



Пример создания веера Ганна:

```
void OnStart()
  {
//---
   string my_gann="OBJ_GANNFAN object";
   if(ObjectFind(0,my_gann)<0)// объект не найден
     {
      //--- сообщим о неудаче
      Print("Object ",my_gann," not found. Error code = ",GetLastError());
      //--- получим максимальную цену графика
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
      //--- получим минимальную цену графика
      double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
      //--- сколько баров на графике показано?
```

© 2000-2025, MetaQuotes Ltd.

```
      int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
      //--- создадим массив, куда запишем время открытия каждого бара
      datetime Time[];
      //--- организуем доступ к массиву как в таймсерии
      ArraySetAsSeries(Time,true);
      //--- теперь скопируем в него данные для видимых на графике баров
      int times=CopyTime(NULL,0,0,bars_on_chart,Time);
      if(times<=0)
        {
         Print("Не удалось скопировать массив со временем открытия!");
         return;
        }
      //--- предварительные приготовления закончены

      //--- индекс центрального бара на графике
      int center_bar=bars_on_chart/2;
      //--- экватор графика - между максимумом и минимумом
      double mean=(chart_max_price+chart_min_price)/2.0;
      //--- установим координаты первой точки привязки в центр
      ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
                   //--- вторая точки привязки справа
                   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
      //--- установим масштаб в единицах Pips/Bar
      ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
      //--- установим направление тренда
      ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
      //--- установим толщину линий
      ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
      //--- зададим стиль линий
      ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
      //--- и цвет линий
      ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
      //--- разрешим пользователю выделять объект
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
      //--- выделим его сами
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
      //--- отрисуем на графике
      ChartRedraw(0);
     }
}
```

© 2000-2025, MetaQuotes Ltd.

Набор Web-цветов

Для типа color определены следующие цветовые константы:

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				

Задавать цвет можно для объектов с помощью функции ObjectSetInteger() и для пользовательских индикаторов с помощью функции PlotIndexSetInteger(). Для получения значения цвета служат аналогичные функции ObjectGetInteger() и PlotIndexGetInteger().

© 2000-2025, MetaQuotes Ltd.

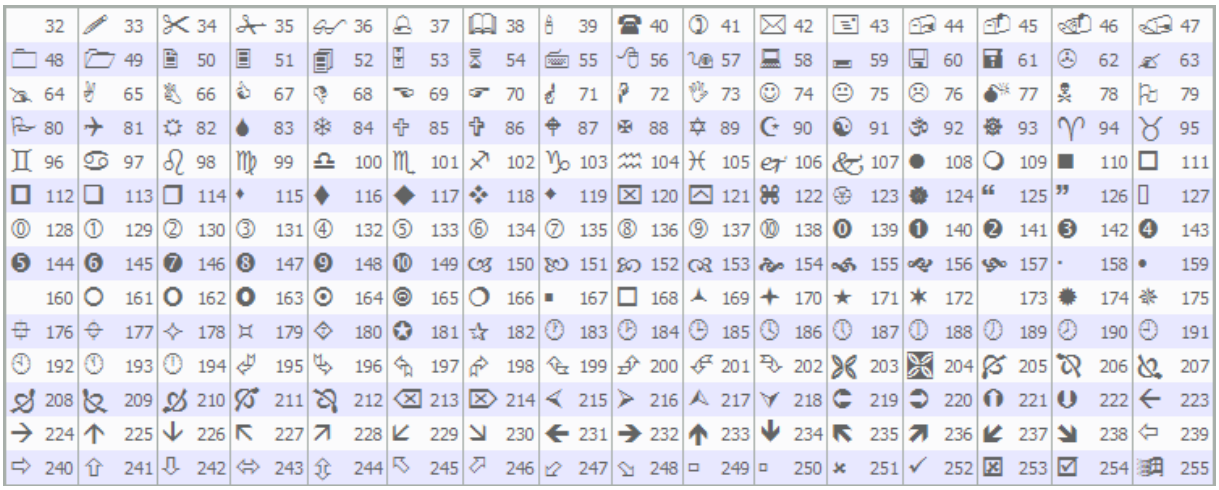
Пример:

```
//---- indicator settings
#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
```

© 2000-2025, MetaQuotes Ltd.

Wingdings

Символы шрифта Wingdings, используемые с объектом OBJ_ARROW:



Установка нужного символа производится с помощью функции ObjectSetInteger().

Пример:

```
void OnStart()
  {
//---
   string up_arrow="up_arrow";
   datetime time=TimeCurrent();
   double lastClose[1];
   int close=CopyClose(Symbol(),Period(),0,1,lastClose);         // получим цену Close
//--- если цена получена
   if(close>0)
     {
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);            // создадим стрелку
      ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);      // установим код стрелки
      ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);          // зададим время
      ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);// зададим цену
      ChartRedraw(0);                                          // перерисуем окно
     }
   else
      Print("Не удалось получить последнюю цену Close!");
  }
```

© 2000-2025, MetaQuotes Ltd.

Константы индикаторов

Предопределены 37 стандартных технических индикаторов, которые можно использовать в программах на языке MQL5. Кроме того, есть возможность создавать свои собственные пользовательские индикаторы с помощью функции iCustom(). Все необходимые для этого константы разделены на 5 групп:

- Ценовые константы - для выбора типа цены или объема, по которым считается индикатор;
- Методы скользящих - встроенные методы сглаживания, используемые в индикаторах;
- Линии индикаторов - идентификаторы индикаторных буферов при доступе к значениям индикаторов с помощью функции CopyBuffer();
- Стили рисования - для указания одного из 18 видов отрисовки и задания стиля рисования линии;
- Свойства пользовательских индикаторов - используются в функциях для работы с пользовательскими индикаторами;
- Типы индикаторов - служат для указания типа технического индикатора при создании хэндла с помощью функции IndicatorCreate();
- Идентификаторы типов данных - используются для задания типа данных, передаваемых массивом типа MqlParam в функции IndicatorCreate().

© 2000-2025, MetaQuotes Ltd.

Ценовые константы

Технические индикаторы требуют для своих расчетов указания значений цен и/или значений объемов, на которых они будут считаться. Существуют 7 предопределенных идентификаторов перечисления ENUM_APPLIED_PRICE, для указания нужной ценовой базы расчетов.

ENUM_APPLIED_PRICE

Идентификатор	Описание
PRICE_CLOSE	Цена закрытия
PRICE_OPEN	Цена открытия
PRICE_HIGH	Максимальная за период цена
PRICE_LOW	Минимальная за период цена
PRICE_MEDIAN	Медианная цена, (high+low)/2
PRICE_TYPICAL	Типичная цена, (high+low+close)/3
PRICE_WEIGHTED	Средневзвешенная цена, (high+low+close+close)/4

Если в расчетах используется объем, то нужно указать одно из двух значений перечисления ENUM_APPLIED_VOLUME.

ENUM_APPLIED_VOLUME

Идентификатор	Описание
VOLUME_TICK	Тиковый объем
VOLUME_REAL	Торговый объем

Технический индикатор iStochastic() имеет два варианта расчета, которые могут использовать:

- либо только цены Close;
- либо цены High и Low.

Для выбора нужного варианта расчета нужно указать одно из значений перечисления ENUM_STO_PRICE.

ENUM_STO_PRICE

Идентификатор	Описание
STO_LOWHIGH	Построение по ценам Low/High
STO_CLOSECLOSE	Построение по ценам Close/Close

Если технический индикатор для своих расчетов использует ценовые данные, тип которых задается перечислением ENUM_APPLIED_PRICE, то в качестве входного ценового ряда можно указывать хэндл любого индикатора (встроенного в терминал или написанного пользователем). В этом случае для расчетов будут использованы значения нулевого буфера индикатора. Это

© 2000-2025, MetaQuotes Ltd.

позволяет легко строить значения одного индикатора по значениям другого индикатора. Хэндл пользовательского индикатора создается вызовом функции iCustom().

Пример:

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- input parameters
input int      RSIperiod=14;         // период для вычисления RSI
input int      Smooth=8;             // период сглаживания RSI
input ENUM_MA_METHOD meth=MODE_SMMA; // метод сглаживания
//---- plot RSI
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- plot 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
//--- indicator buffers
double         RSIBuffer[];          // здесь мы будем хранить значения RSI
double         RSI_SmoothedBuffer[]; // здесь будут сглаженные значения RSI
int            RSIhandle;            // дескриптор на индикатор RSI
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   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);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[]
```

© 2000-2025, MetaQuotes Ltd.

```
                  )

  {
//--- сбросим в ноль значение последней ошибки
   ResetLastError();
//--- получим данные индикатора RSI в массив RSIBuffer[]
   int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
   if(copied<=0)
     {
      Print("Не удалось скопировать значения индикатора RSI. Error = ",
            GetLastError(),",  copied = ",copied);
      return(0);
     }
//--- создадим индикатор средней по значениям индикатора 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("Не удалось скопировать сглаженный индикатор RSI. Error = ",
            GetLastError(),",  copied =",copied);
      return(0);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

Методы скользящих

В основе многих технических индикаторов лежат те или иные методы сглаживания ценовых серий. Некоторые стандартные технические индикаторы требуют указания типа сглаживания в качестве параметра. Для указания нужного типа сглаживания служат идентификаторы, перечисленные в перечислении ENUM_MA_METHOD.

ENUM_MA_METHOD

Идентификатор	Описание
MODE_SMA	Простое усреднение
MODE_EMA	Экспоненциальное усреднение
MODE_SMMA	Сглаженное усреднение
MODE_LWMA	Линейно-взвешенное усреднение

Пример:

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- handles for moving averages
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- get MA's handles
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);
```

© 2000-2025, MetaQuotes Ltd.

Линии индикаторов

Некоторые технические индикаторы имеют несколько отрисовываемых на графике буферов. Нумерация индикаторных буферов начинается с 0. При копировании значений индикатора функцией CopyBuffer() в массив типа double для некоторых индикаторов можно указывать не номер копируемого буфера, а идентификатор этого буфера.

Идентификаторы линий индикаторов, допустимых при копировании значений индикаторов iMACD(), iRVI() и iStochastic()

Константа	Значение	Описание
MAIN_LINE	0	Основная линия
SIGNAL_LINE	1	Сигнальная линия

Идентификаторы линий индикаторов, допустимых при копировании значений индикаторов ADX() и ADXW()

Константа	Значение	Описание
MAIN_LINE	0	Основная линия
PLUSDI_LINE	1	Линия +DI
MINUSDI_LINE	2	Линия -DI

Идентификаторы линий индикаторов допустимых при копировании значений индикатора iBands()

Константа	Значение	Описание
BASE_LINE	0	Основная линия
UPPER_BAND	1	Верхняя граница
LOWER_BAND	2	Нижняя граница

Идентификаторы линий индикаторов, допустимых при копировании значений индикаторов iEnvelopes() и iFractals()

Константа	Значение	Описание
UPPER_LINE	0	Верхняя линия
LOWER_LINE	1	Нижняя линия

Идентификаторы линий индикаторов, допустимых при копировании значений индикатора iGator()

Константа	Значение	Описание
UPPER_HISTOGRAM	0	Верхняя гистограмма
LOWER_HISTOGRAM	2	Нижняя гистограмма

Идентификаторы линий индикаторов, допустимых при копировании значений индикатора iAlligator()

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
GATORJAW_LINE	0	Линия челюстей
GATORTEETH_LINE	1	Линия зубов
GATORLIPS_LINE	2	Линия губ

Идентификаторы линий индикаторов, допустимых при копировании значений индикатора iIchimoku()

Константа	Значение	Описание
TENKANSEN_LINE	0	Линия Tenkan-sen
KIJUNSEN_LINE	1	Линия Kijun-sen
SENKOUSPANA_LINE	2	Линия Senkou Span A
SENKOUSPANB_LINE	3	Линия Senkou Span B
CHIKOUSPAN_LINE	4	Линия Chikou Span

© 2000-2025, MetaQuotes Ltd.

Стили рисования

При создании пользовательского индикатора можно указать один из 18 типов графического построения (способа отображения на главном окне графика или в подокне графика), значения которых указаны в перечислении ENUM_DRAW_TYPE.

В одном пользовательском индикаторе допустимо использовать любые виды построения/отрисовки индикаторов. Каждый вид построения требует указания от одного до пяти глобальных массивов для хранения данных, необходимых для рисования. Эти массивы данных необходимо связать с индикаторными буферами посредством функции SetIndexBuffer(), и указать для каждого буфера тип данных из перечисления ENUM_INDEXBUFFER_TYPE.

В зависимости от стиля рисования, может потребоваться от одного до четырех буферов значений (отмеченных как INDICATOR_DATA), а если стиль допускает динамическое чередование цвета (все стили со словом COLOR в наименовании), то необходим и один буфер цвета (указан тип INDICATOR_COLOR_INDEX). Цветовой буфер всегда связывается сразу после соответствующих стилю буферов значений.

ENUM_DRAW_TYPE

Идентификатор	Описание	Буферов значений	Буферов цвета
DRAW_NONE	Не отрисовывается	1	0
DRAW_LINE	Линия	1	0
DRAW_SECTION	Отрезки	1	0
DRAW_HISTOGRAM	Гистограмма от нулевой линии	1	0
DRAW_HISTOGRAM2	Гистограмма на двух индикаторных буферах	2	0
DRAW_ARROW	Отрисовка стрелками	1	0
DRAW_ZIGZAG	Стиль Zigzag допускает вертикальные	2	0

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Буферов значений	Буферов цвета
	отрезки на баре		
DRAW_FILLING	Цветовая заливка между двумя уровнями	2	0
DRAW_BARS	Отображение в виде баров	4	0
DRAW_CANDLES	Отображение в виде свечей	4	0
DRAW_COLOR_LINE	Разноцветная линия	1	1
DRAW_COLOR_SECTION	Разноцветные отрезки	1	1
DRAW_COLOR_HISTOGRAM	Разноцветная гистограмма от нулевой линии	1	1
DRAW_COLOR_HISTOGRAM2	Разноцветная гистограмма на двух индикаторных буферах	2	1
DRAW_COLOR_ARROW	Отрисовка разноцветными стрелками	1	1

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Буферов значений	Буферов цвета
DRAW_COLOR_ZIGZAG	Разноцветный ZigZag	2	1
DRAW_COLOR_BARS	Разноцветные бары	4	1
DRAW_COLOR_CANDLES	Разноцветные свечи	4	1

Для уточнения отображения выбранного вида отрисовки используются идентификаторы, перечисленные в перечислениях ENUM_PLOT_PROPERTY.

Для функций PlotIndexSetInteger() и PlotIndexGetInteger()

ENUM_PLOT_PROPERTY_INTEGER

Идентификатор	Описание	Тип свойства
PLOT_ARROW	Код стрелки для стиля DRAW_ARROW	uchar
PLOT_ARROW_SHIFT	Смещение стрелок по вертикали для стиля DRAW_ARROW	int
PLOT_DRAW_BEGIN	Количество начальных баров без отрисовки и значений в DataWindow	int
PLOT_DRAW_TYPE	Тип графического построения	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	Признак отображения значений построения в окне DataWindow	bool
PLOT_SHIFT	Сдвиг графического	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	построения индикатора по оси времени в барах	
PLOT_LINE_STYLE	Стиль линии отрисовки	ENUM_LINE_STYLE
PLOT_LINE_WIDTH	Толщина линии отрисовки	int
PLOT_COLOR_INDEXES	Количество цветов	int
PLOT_LINE_COLOR	Цвет отрисовки	color модификатор=номер индекса цвета

Для функции PlotIndexSetDouble()

ENUM_PLOT_PROPERTY_DOUBLE

Идентификатор	Описание	Тип свойства
PLOT_EMPTY_VALUE	Пустое значение для построения, для которого нет отрисовки	double

Для функции PlotIndexSetString()

ENUM_PLOT_PROPERTY_STRING

Идентификатор	Описание	Тип свойства
PLOT_LABEL	Имя индикаторной графической серии для отображения в окне DataWindow. Для сложных графических стилей, требующих для отображения несколько индикаторных буферов, имена для каждого буфера можно задать с	string

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	использованием ";" в качестве разделителя. Пример кода приведен в DRAW_CANDLES	

Существуют 5 стилей, которыми может рисоваться линия в пользовательском индикаторе. Они применимы только при толщине линии 0 или 1.

ENUM_LINE_STYLE

Идентификатор	Описание
STYLE_SOLID	Сплошная линия
STYLE_DASH	Прерывистая линия
STYLE_DOT	Пунктирная линия
STYLE_DASHDOT	Штрих-пунктирная линия
STYLE_DASHDOTDOT	Штрих - две точки

Для задания стиля рисования линии и вида отрисовки используется функция PlotIndexSetInteger(). Для расширений Фибоначчи указать толщину и стиль отрисовки уровней можно функцией ObjectSetInteger().

Пример:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- indicator buffers
double         MABuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- привязка массива к индикаторному буферу с индексом 0
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- задать рисование линии
   PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- задание стиля для рисования линии
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- задание цвета линии
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- задание толщины линии
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- задание метки для линии
   PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//---
   for(int i=prev_calculated;i<rates_total;i++)
      MABuffer[i]=close[i];
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

Свойства пользовательских индикаторов

Количество индикаторных буферов, которые можно использовать в пользовательском индикаторе, не ограничено. Но каждому массиву, который назначается в качестве индикаторного буфера с помощью функции SetIndexBuffer(), должен быть указан тип данных, которые он будет хранить. Это может быть одно из значений перечисления ENUM_INDEXBUFFER_TYPE.

ENUM_INDEXBUFFER_TYPE

Идентификатор	Описание
INDICATOR_DATA	Данные для отрисовки
INDICATOR_COLOR_INDEX	Цвета отрисовки
INDICATOR_CALCULATIONS	Вспомогательные буферы для промежуточных вычислений

Пользовательский индикатор имеет множество настроек для удобства отображения и восприятия. Эти настройки производятся через задание соответствующих свойств индикатора с помощью функций IndicatorSetDouble(), IndicatorSetInteger() и IndicatorSetString(). Идентификаторы свойств индикатора перечислены в перечислениях ENUM_CUSTOMIND_PROPERTY.

ENUM_CUSTOMIND_PROPERTY_INTEGER

Идентификатор	Описание	Тип свойства
INDICATOR_DIGITS	Точность отображения значений индикатора	int
INDICATOR_HEIGHT	Фиксированная высота собственн	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства	
	ого окна индикатора (команда препроцессора #property indicator_height)		
INDICATOR_LEVELS	Количество уровней на окне индикатора	int	
INDICATOR_LEVELCOLOR	Цвет линии уровня	color	модификатор=номер уровня

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
INDICATOR_LEVELSTYLE	Стиль линии уровня	ENUM_LINE_STYLE модификатор=номер уровня
INDICATOR_LEVELWIDTH	Толщина линии уровня	int модификатор=номер уровня
INDICATOR_FIXED_MINIMUM	Фиксированный минимум для окна индикатора. Свойство доступно только на запись функцией IndicatorS	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	etlinteger()	
INDICATOR_FIXED_MAXIMUM	Фиксированный максимум для окна индикатора. Свойство доступно только на запись функцией IndicatorSetInteger()	bool

ENUM_CUSTOMIND_PROPERTY_DOUBLE

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства	
INDICATOR_MINIMUM	Минимум окна индикатора	double	
INDICATOR_MAXIMUM	Максимум окна индикатора	double	
INDICATOR_LEVELVALUE	Значение уровня	double	модификатор=номер уровня

ENUM_CUSTOMIND_PROPERTY_STRING

Идентификатор	Описание	Тип свойства	
INDICATOR_SHORTNAME	Короткое наименование индикатора	string	
INDICATOR_LEVELTEXT	Описа	string	модификатор=номер уровня

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ние уровня	

Примеры:

```
//--- indicator settings
#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
//--- input parameters
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- indicator buffers
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
double LowesBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
   SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- set accuracy
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- set levels
   IndicatorSetInteger(INDICATOR_LEVELS,2);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- set maximum and minimum for subwindow
   IndicatorSetDouble(INDICATOR_MINIMUM,0);
   IndicatorSetDouble(INDICATOR_MAXIMUM,100);
//--- sets first bar from what index will be drawn
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- set style STYLE_DOT for second line
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- name for DataWindow and indicator subwindow label
   IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+","+DPeriod+","+Slowing+")"
   PlotIndexSetString(0,PLOT_LABEL,"Main");
   PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- sets drawing line to empty value
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- initialization done
  }
```

© 2000-2025, MetaQuotes Ltd.

Типы технических индикаторов

Существует два способа программно создавать хэндл индикатора для последующего доступа к его значениям. Первый способ состоит в непосредственном указании имени функции из списка технических индикаторов. Второй способ позволяет с помощью функции IndicatorCreate() единообразно создавать хэндл любого индикатора заданием идентификатора из перечисления ENUM_INDICATOR. Оба варианта создания хэндла индикатора равноправны, можно использовать тот, который наиболее удобен в каждом конкретном случае при написании программы на MQL5.

При создании индикатора типа IND_CUSTOM, поле *type* первого элемента массива входных параметров MqlParam обязательно должен иметь значение TYPE_STRING из перечисления ENUM_DATATYPE, а поле *string_value* первого элемента должно содержать имя пользовательского индикатора.

ENUM_INDICATOR

Идентификатор	Индикатор
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

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Индикатор
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

© 2000-2025, MetaQuotes Ltd.

Идентификаторы типов данных

При создании хэндла индикатора функцией IndicatorCreate() необходимо указывать последним параметром массив типа MqlParam. Соответственно, структура MqlParam, описывающая параметры индикатора, содержит специальное поле *type*. Это поле содержит информацию о типе данных (вещественный, целочисленный или строковый тип), которые передаются конкретным элементом этого массива. Значение этого поля структуры MqlParam может быть одним из значений перечисления ENUM_DATATYPE.

ENUM_DATATYPE

Идентификатор	Тип данных
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

Каждый элемент этого массива описывает соответствующий входной параметр создаваемого технического индикатора, поэтому тип и порядок следования элементов массива должен быть строго выдержан в соответствии с описанием.

© 2000-2025, MetaQuotes Ltd.

Состояние окружения

Константы, описывающие текущую среду выполнения mql5-программы делятся на группы:

- Состояние клиентского терминала - информация о клиентском терминале;
- Информация о запущенной MQL5-программе - свойства mql5-программы, которые помогают дополнительно управлять ее поведением;
- Информация об инструменте - получение торговой информации об инструменте;
- Информация о счете - информация о текущем торговом счете;
- Статистика тестирования - результаты тестирования эксперта.

© 2000-2025, MetaQuotes Ltd.

Состояние клиентского терминала

Идентификаторы для получения информации о клиентском терминале функциями TerminalInfoInteger() и TerminalInfoString(). В качестве параметра эти функции принимают значения из перечислений ENUM_TERMINAL_INFO_INTEGER и ENUM_TERMINAL_INFO_STRING соответственно.

ENUM_TERMINAL_INFO_INTEGER

Идентификатор	Описание	Тип свойства
TERMINAL_BUILD	Номер билда запущенного терминала	int
TERMINAL_COMMUNITY_ACCOUNT	Флаг наличия авторизационных данных MQL5.community в терминале	bool
TERMINAL_COMMUNITY_CONNECTION	Наличие подключения к MQL5.community	bool
TERMINAL_CONNECTED	Наличие подключения к торговому серверу	bool
TERMINAL_DLLS_ALLOWED	Разрешение на использование DLL	bool
TERMINAL_TRADE_ALLOWED	Разрешение на торговлю	bool
TERMINAL_EMAIL_ENABLED	Разрешение на отправку писем с использованием SMTP-сервера и логина, указанных в настройках терминала	bool
TERMINAL_FTP_ENABLED	Разрешение на отправку отчетов по FTP на указанный сервер для указанного в настройках терминала торгового счета	bool
TERMINAL_NOTIFICATIONS_ENABLED	Разрешение на отправку уведомлений на смартфон	bool
TERMINAL_MAXBARS	Максимальное количество баров на графике	int
TERMINAL_MQID	Флаг наличия MetaQuotes ID для отправки Push-уведомлений	bool
TERMINAL_CODEPAGE	Номер кодовой страницы языка, установленного в клиентском терминале	int
TERMINAL_CPU_CORES	Количество процессоров в системе	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
TERMINAL_DISK_SPACE	Объем свободной памяти на диске для папки MQL5\Files терминала (агента), в MB	int
TERMINAL_MEMORY_PHYSICAL	Размер физической памяти в системе, в MB	int
TERMINAL_MEMORY_TOTAL	Размер памяти, доступной процессу терминала (агента), в MB	int
TERMINAL_MEMORY_AVAILABLE	Размер свободной памяти процесса терминала (агента), в MB	int
TERMINAL_MEMORY_USED	Размер памяти, использованной терминалом (агентом), в MB	int
TERMINAL_X64	Признак "64 битный терминал"	bool
TERMINAL_OPENCL_SUPPORT	Версия поддерживаемой OpenCL в виде 0x00010002 = 1.2. "0" означает, что OpenCL не поддерживается	int
TERMINAL_SCREEN_DPI	Разрешающая способность вывода информации на экран измеряется в количестве точек на линейный дюйм поверхности (DPI). Знание этого параметра позволяет задавать размеры графических объектов таким образом, чтобы они выглядели одинаково на мониторах с различной разрешающей способностью.	int
TERMINAL_SCREEN_LEFT	Левая координата виртуального экрана. Виртуальным экраном является прямоугольник, охватывающий все мониторы. Если в системе имеется два монитора и их порядок задан справа налево, то левая координата виртуального экрана может оказаться на границе двух мониторов.	int
TERMINAL_SCREEN_TOP	Верхняя координата виртуального экрана	int
TERMINAL_SCREEN_WIDTH	Ширина терминала	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
TERMINAL_SCREEN_HEIGHT	Высота терминала	int
TERMINAL_LEFT	Левая координата терминала относительно виртуального экрана	int
TERMINAL_TOP	Верхняя координата терминала относительно виртуального экрана	int
TERMINAL_RIGHT	Правая координата терминала относительно виртуального экрана	int
TERMINAL_BOTTOM	Нижняя координата терминала относительно виртуального экрана	int
TERMINAL_PING_LAST	Последнее известное значение пинга до торгового сервера в микросекундах. В одной секунде миллион микросекунд	int
TERMINAL_VPS	Признак того, что терминал запущен на виртуальном сервере MetaTrader Virtual Hosting (MetaTrader VPS)	bool
Идентификатор клавиши	**Описание**	
TERMINAL_KEYSTATE_LEFT	Состояние клавиши "Стрелка влево"	int
TERMINAL_KEYSTATE_UP	Состояние клавиши "Стрелка вверх"	int
TERMINAL_KEYSTATE_RIGHT	Состояние клавиши "Стрелка вправо"	int
TERMINAL_KEYSTATE_DOWN	Состояние клавиши "Стрелка вниз"	int
TERMINAL_KEYSTATE_SHIFT	Состояние клавиши "Shift"	int
TERMINAL_KEYSTATE_CONTROL	Состояние клавиши "Ctrl"	int
TERMINAL_KEYSTATE_MENU	Состояние клавиши "Windows"	int
TERMINAL_KEYSTATE_CAPSLOCK	Состояние клавиши "CapsLock"	int
TERMINAL_KEYSTATE_NUMLOCK	Состояние клавиши "NumLock"	int
TERMINAL_KEYSTATE_SCRLOCK	Состояние клавиши "ScrollLock"	int
TERMINAL_KEYSTATE_ENTER	Состояние клавиши "Enter"	int
TERMINAL_KEYSTATE_INSERT	Состояние клавиши "Insert"	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
TERMINAL_KEYSTATE_DELETE	Состояние клавиши "Delete"	int
TERMINAL_KEYSTATE_HOME	Состояние клавиши "Home"	int
TERMINAL_KEYSTATE_END	Состояние клавиши "End"	int
TERMINAL_KEYSTATE_TAB	Состояние клавиши "Tab"	int
TERMINAL_KEYSTATE_PAGEUP	Состояние клавиши "PageUp"	int
TERMINAL_KEYSTATE_PAGEDOWN	Состояние клавиши "PageDown"	int
TERMINAL_KEYSTATE_ESCAPE	Состояние клавиши "Escape"	int

Вызов TerminalInfoInteger(TERMINAL_KEYSTATE_XXX) возвращает такой же код состояния клавиши, как и функция GetKeyState() из MSDN.

Пример вычисления коэффициента масштабирования:

```
//--- создаём кнопку шириной 1.5 дюйма на экране
int screen_dpi = TerminalInfoInteger(TERMINAL_SCREEN_DPI); // получим DPI монитора пол
int base_width = 144;                                      // базовая ширина в экранны
int width      = (button_width * screen_dpi) / 96;         // вычислим ширину кнопки д
...

//--- вычисление коэффициента масштабирования в процентах
int scale_factor=(TerminalInfoInteger(TERMINAL_SCREEN_DPI) * 100) / 96;
//--- использование коэффициента масштабирования
width=(base_width * scale_factor) / 100;
```

При таком использовании графический ресурс будет иметь одинаковый на глаз размер на мониторах с различной разрешающей способностью. При этом размеры управляющих элементов (кнопки, окна диалогов и т.д.) будут соответствовать настройкам персонализации.

ENUM_TERMINAL_INFO_DOUBLE

Идентификатор	Описание	Тип свойства
TERMINAL_COMMUNITY_BALANCE	Баланс пользователя в MQL5.community	double
TERMINAL_RETRANSMISSION	Процент повторно отправляемых сетевых пакетов в TCP/IP протоколе для всех запущенных приложений и служб на данном компьютере. Даже в самой быстрой и правильно настроенной сети происходят потери пакетов и, как	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	следствие, отсутствие подтверждений о доставке пакетов между получателем и отправителем. В таких случаях производится повторная отправка "потерянного" пакета. Не является показателем качества подключения конкретного терминала к конкретному торговому серверу, так как считается для всей сетевой активности, включая системную и фоновую. Показатель TERMINAL_RETRANSMISSION запрашивается раз в минуту из операционной системы. Сам терминал не считает этот показатель.	

Файловые операции можно проводить только в двух каталогах, пути к которым можно получить при запросе свойств TERMINAL_DATA_PATH и TERMINAL_COMMONDATA_PATH.

ENUM_TERMINAL_INFO_STRING

Идентификатор	Описание	Тип свойства
TERMINAL_LANGUAGE	Язык терминала	string
TERMINAL_COMPANY	Имя компании	string
TERMINAL_NAME	Имя терминала	string
TERMINAL_PATH	Папка, из которой запущен терминал	string
TERMINAL_DATA_PATH	Папка, в которой хранятся данные терминала	string
TERMINAL_COMMONDATA_PATH	Общая папка всех клиентских терминалов, установленных на компьютере	string
TERMINAL_CPU_NAME	Наименование процессора	string
TERMINAL_CPU_ARCHITECTURE	Архитектура процессора	string
TERMINAL_OS_VERSION	Наименование операционной системы пользователя	string

© 2000-2025, MetaQuotes Ltd.

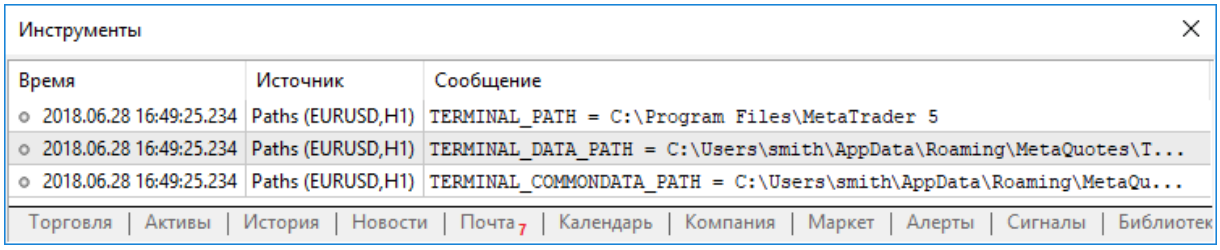
Идентификатор	Описание	Тип свойства
TERMINAL_COLORTHEME_NAME	Цветовая схема терминала, возможные значения: Light и Dark.	string

Для лучшего понимания путей, хранящихся в свойствах параметров TERMINAL_PATH, TERMINAL_DATA_PATH и TERMINAL_COMMONDATA_PATH, рекомендуется выполнить скрипт, которых сообщит эти значения для данной копии терминала, установленного на вашем компьютере.

Пример: скрипт выводит информацию о путях терминала

```
//+------------------------------------------------------------------+
//|                                          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"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
   Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
   Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
  }
```

В результате выполнения скрипта в Журнал Экспертов будут выведены сообщения, подобные приведенным на рисунке ниже.



Получение информации о цветовой схеме терминала

Терминал позволяет задавать цветовые схемы — светлую (по умолчанию) и темную. При создании пользовательских приложений, которые имеют графические интерфейс для взаимодействия с

© 2000-2025, MetaQuotes Ltd.

пользователем, разработчикам необходимо учитывать текущую цветовую схему терминала. Следовательно, требуется динамическая адаптация использованных в приложении визуальных компонентов для удобства пользователей.

С этой целью добавлены функции для определения цветовой схемы терминала:

- Значение TERMINAL_COLORTHEME_NAME из перечислении ENUM_TERMINAL_INFO_STRING позволяет получить название цветовой схемы при помощи функции TerminalInfoString. Возможные значения: Light и Dark.
- Используйте свойства THEME_COLOR_* из перечислении ENUM_TERMINAL_INFO_INTEGER для получения цветов конкретных элементов интерфейса при помощи функции TerminalInfoInteger.

Идентификатор	Описание	Тип свойства
THEME_COLOR_WINDOW	Фон окна	color
THEME_COLOR_WINDOWTEXT	Текст в окне	color
THEME_COLOR_BTNTEXT	Текст на кнопках	color
THEME_COLOR_GRAYTEXT	Неактивный (отключённый) текст	color
THEME_COLOR_INFOTEXT	Текст в подсказках	color
THEME_COLOR_INFOBK	Фон подсказок	color
THEME_COLOR_3DFACE	Лицевая сторона 3D-элементов	color
THEME_COLOR_3DLIGHT	Светлая сторона 3D-элементов	color
THEME_COLOR_3DSHADOW	Теневая сторона 3D-элементов	color
THEME_COLOR_3DDKSHADOW	Тёмная тень 3D-элементов	color
THEME_COLOR_3DHILIGHT	Подсветка 3D-элементов	color
THEME_COLOR_HIGHLIGHT	Фон выделенных элементов	color
THEME_COLOR_HIGHLIGHTTEXT	Текст выделенных элементов	color
THEME_COLOR_BTNFACE	Лицевая сторона кнопок	color
THEME_COLOR_BTNHILIGHT	Подсветка кнопок	color
THEME_COLOR_BTNSHADOW	Тень кнопок	color
THEME_COLOR_MENU	Фон меню	color
THEME_COLOR_MENUBAR	Фон панели меню	color
THEME_COLOR_MENUTEXT	Текст меню	color
THEME_COLOR_MENUHILIGHT	Подсветка выбранного пункта меню	color
THEME_COLOR_ACTIVECAPTION	Заголовок активного окна	color
THEME_COLOR_INACTIVECAPTION	Заголовок неактивного окна	color

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
THEME_COLOR_GRADIENTINACTIVECAPTION	Градиент заголовка неактивного окна	color
THEME_COLOR_CAPTIONTEXT	Текст заголовка окна	color
THEME_COLOR_INACTIVECAPTIONTEXT	Текст заголовка неактивного окна	color
THEME_COLOR_HOTTEXT	Гиперссылки или активные элементы	color
THEME_COLOR_NONE	Цвет не выбран	color
THEME_COLOR_SEPARATOR	Разделитель	color
THEME_COLOR_SCROLLBACK	Скроллбар	color
THEME_COLOR_LINE1	Цвет фона нечетной строки в Журнале	color
THEME_COLOR_LINE2	Цвет фона четной строки в Журнале	color
THEME_COLOR_GRID	Цвет сетки в Журнале	color
THEME_COLOR_SUMMARY	Цвет фона итоговой строки в Журнале	color
THEME_COLOR_ERROR	Цвет текста об ошибки	color
THEME_COLOR_INVALID	Цвет текста о невалидном значении	color
THEME_COLOR_NEGATIVE	Цвет отрицательного значения	color
THEME_COLOR_POSITIVE	Цвет положительного значения	color
THEME_COLOR_LINK	Цвет ссылки	color
THEME_COLOR_LINKHOVER	Цвет ссылки при наведении мышки	color
THEME_COLOR_LINKTESTER	Цвет ссылок в кеше результатов предыдущих запусков тестирования/оптимизации	color
THEME_COLOR_TEXTUP	Кнопка отжата	color
THEME_COLOR_TEXTDOWN	Кнопка нажата	color
THEME_COLOR_BACKUP	Цвет кнопок "BUY" и "SELL" в окне "One Click Trading" при повышении предложенной цены	color

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
THEME_COLOR_BACKDOWN	Цвет кнопок "BUY" и "SELL" в окне "One Click Trading" при понижении предложенной цены	color
THEME_COLOR_CLOSE	Кнопка операции "Close"	color
THEME_COLOR_BUY	Кнопка операции "BUY"	color
THEME_COLOR_SELL	Кнопка операции "SELL"	color
THEME_COLOR_DEPOSIT	Цвет кнопки "Deposit"	color
THEME_COLOR_WITHDRAWAL	Цвет кнопки "Withdrawal"	color
THEME_COLOR_BID	Цвет линии "Bid"	color
THEME_COLOR_ASK	Цвет линии "Ask"	color
THEME_COLOR_STOPS	Цвет линии "Stop"	color
THEME_COLOR_STOPS_RED	Подсветка значения StopLoss при отрицательном значении прибыли на вкладке Торговля	color
THEME_COLOR_STOPS_GREEN	Подсветка значения StopLoss при положительном значении прибыли на вкладке Торговля	color
THEME_COLOR_CONFIRM	Цвет кнопки "Accept" в окне отправки ордера	color
THEME_COLOR_REQUOTE	Цвет кнопки "Requote" в окне отправки ордера	color
THEME_COLOR_REJECT	Цвет кнопки "Reject" в окне отправки ордера	color
THEME_COLOR_NOTIFICATION	Цвет текста об изменении от сервера в окне отправки ордера	color
THEME_COLOR_RATING	Цвет полоски рейтинга в системе обучения	color
THEME_COLOR_BOOK_BUY	Цвет фона уровней на покупку в Стакане цен	color
THEME_COLOR_BOOK_SELL	Цвет фона уровней на продажу в Стакане цен	color
THEME_COLOR_BOOK_LAST	Цвет последней сделки в Стакане цен	color
THEME_COLOR_BOOK_STOP	Цвет уровня StopLoss в Стакане цен	color

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
THEME_COLOR_BOOK_SPREAD	Цвет фона уровней внутри спреда в Стакане цен	color
THEME_COLOR_TICKS_BID	Цвет линии Bid тикового графика в окне отправки ордера	color
THEME_COLOR_TICKS_ASK	Цвет линии Ask тикового графика в окне отправки ордера	color
THEME_COLOR_TICKS_LAST	Цвет линии Last тикового графика в окне отправки ордера	color
THEME_COLOR_TICKS_CROSS	Цвет перекрестья на тиковом графике в окне отправки ордера	color
THEME_COLOR_TICKS_SL	Цвет линии StopLoss на тиковом графике в окне отправки ордера	color
THEME_COLOR_TICKS_TP	Цвет линии TakeProfit на тиковом графике в окне отправки ордера	color
THEME_COLOR_TESTER_START	Цвет кнопки "Старт" тестирования/оптимизации	color
THEME_COLOR_TESTER_STOP	Цвет кнопки "Стоп" тестирования/оптимизации	color
THEME_COLOR_TESTER_START_FRAME	Цвет рамки кнопки "Старт" тестирования/оптимизации	color
THEME_COLOR_TESTER_STOP_FRAME	Цвет рамки кнопки "Стоп" тестирования/оптимизации	color
THEME_COLOR_TESTER_PROGRESS	Цвет прогресс-бара тестирования/оптимизации	color
THEME_COLOR_TESTER_BALANCE	Цвет линии Баланс в тестере стратегий	color
THEME_COLOR_TESTER_EQUITY	Цвет линии Эквити в тестере стратегий	color
THEME_COLOR_TESTER_MARGIN	Цвет графика загрузки депозита в тестере стратегий	color
THEME_COLOR_PROFILER_CALL	Цвет строки кода с вызовом при Профилировке	color

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
THEME_COLOR_PROFILER_CALLSEL	Цвет выделенной строки кода с вызовом при Профилировки	color
THEME_COLOR_PROFILER_LINE	Цвет строки в Журнале Профилировки	color
THEME_COLOR_PROFILER_LINESEL	Цвет выделенной строки в Журнале Профилировки	color

© 2000-2025, MetaQuotes Ltd.

Информация о запущенной MQL5-программе

Константы, для получения информации о выполняющейся mql5-программе, перечислены в ENUM_MQL_INFO_INTEGER и ENUM_MQL_INFO_STRING.

Для функции MQLInfoInteger()

ENUM_MQL_INFO_INTEGER

Идентификатор	Описание	Тип свойства
MQL_HANDLES_USED	Текущее количество активных хендлов объектов. Учитываются как динамические (созданные через new), так и нединамические объекты, глобальные/локальные переменные или члены классов. Чем больше хэндлов использует программа, тем больше она потребляет ресурсов.	int
MQL_MEMORY_LIMIT	Максимально возможный объём динамической памяти для MQL5-программы в MB	int
MQL_MEMORY_USED	Размер использованной памяти MQL5-программой в MB	int
MQL_PROGRAM_TYPE	Тип mql5-программы	ENUM_PROGRAM_TYPE

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
MQL_DLLS_ALLOWED	Разрешение на использование DLL для **данной запущенной программы**	bool
MQL_TRADE_ALLOWED	Разрешение на торговлю для **данной запущенной программы**	bool
MQL_SIGNALS_ALLOWED	Разрешение на работу с сигналами **данной запущенной программы**	bool
MQL_DEBUG	Признак работы запущенной программы в режиме отладки	bool
MQL_PROFILER	Признак работы запущенной программы в режиме профилирования кода	bool
MQL_TESTER	Признак работы запущенной программы в тестере	bool
MQL_FORWARD	Признак работы запущенной программы в процессе форвардного тестирования	bool
MQL_OPTIMIZATION	Признак работы запущенной программы в процессе оптимизации	bool
MQL_VISUAL_MODE	Признак работы запущенной программы в	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	визуальном режиме тестирования	
MQL_FRAME_MODE	Признак работы запущенного эксперта на графике в режиме сбора фреймов результатов оптимизации	bool
MQL_LICENSE_TYPE	Тип лицензии модуля EX5. Лицензия относится именно к тому модулю EX5, из которого делается запрос с помощью MQLInfoInteger(MQL_LICENSE_TYPE).	ENUM_LICENSE_TYPE
MQL_STARTED_FROM_CONFIG	Возвращает true, если скрипт/эксперт запущен из секции StartUp конфигурационного файла. Это означает, что данный скрипт/эксперт был прописан в конфигурационном файле, с которым запущен терминал.	bool

Для функции MQLInfoString()

ENUM_MQL_INFO_STRING

Идентификатор	Описание	Тип свойства
MQL_PROGRAM_NAME	Имя запущенной MQL5-	string

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	программы	
MQL_PROGRAM_PATH	Путь для данной запущенной программы	string

Для получения информации о типе выполняемой программы предназначены значения перечисления ENUM_PROGRAM_TYPE.

ENUM_PROGRAM_TYPE

Идентификатор	Описание
PROGRAM_SCRIPT	Скрипт
PROGRAM_EXPERT	Эксперт
PROGRAM_INDICATOR	Индикатор
PROGRAM_SERVICE	Сервис

ENUM_LICENSE_TYPE

Идентификатор	Описание
LICENSE_FREE	Бесплатная неограниченная версия
LICENSE_DEMO	Демо-версия платного продукта из Маркета. Работает только в тестере стратегий
LICENSE_FULL	Купленная лицензионная версия допускает не менее 5 активаций. Продавец может увеличить разрешенное число активаций
LICENSE_TIME	Версия с ограниченной по времени лицензией

Пример:

```
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");
```

© 2000-2025, MetaQuotes Ltd.

```
         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), " hanc
```

© 2000-2025, MetaQuotes Ltd.

Информация об инструменте

Для получения текущей рыночной информации служат функции SymbolInfoInteger(), SymbolInfoDouble() и SymbolInfoString(). В качестве второго параметра этих функций допустимо передавать один из идентификаторов из перечислений ENUM_SYMBOL_INFO_INTEGER, ENUM_SYMBOL_INFO_DOUBLE и ENUM_SYMBOL_INFO_STRING соответственно.

Для функции SymbolInfoInteger()

ENUM_SYMBOL_INFO_INTEGER

Идентификатор	Описание	Тип свойства
SYMBOL_SUBSCRIPTION_DELAY	Признак того, что данные по символу идут с задержкой. Свойство можно запрашивать только для символов, выбранных в MarketWatch (SYMBOL_SELECT = true). Для невыбранных символов будет сгенерирована ошибка ERR_MARKET_NOT_SELECTED (4302)	bool
SYMBOL_SECTOR	Сектор экономики, к которому относится символ	ENUM_SYMBOL_SECTOR
SYMBOL_INDUSTRY	Вид промышленности или отрасль экономики, к которому относится символ	ENUM_SYMBOL_INDUSTRY
SYMBOL_CUSTOM	Признак того, что символ является пользовательским, то есть искусственно созданным на основе других символов из Market Watch или/и внешних источников данных	bool
SYMBOL_BACKGROUND_COLOR	Цвет фона, которым подсвечивается символ в Market Watch	color

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
SYMBOL_CHART_MODE	Тип цены для построения баров - Bid или Last	ENUM_SYMBOL_CHART_MODE
SYMBOL_EXIST	Признак того, что символ с таким именем существует	bool
SYMBOL_SELECT	Признак того, что символ выбран в Market Watch	bool
SYMBOL_VISIBLE	Признак того, что выбранный символ отображается в Market Watch. Некоторые символы (как правило, это кросс-курсы, которые необходимы для расчёта маржевых требований и прибыли в валюте депозита) выбираются автоматически, при этом могут не отображаться в Market Watch. Для отображения такие символы должны быть выбраны явно.	bool
SYMBOL_SESSION_DEALS	Количество сделок в текущей сессии	long
SYMBOL_SESSION_BUY_ORDERS	Общее число ордеров на покупку в текущий момент	long
SYMBOL_SESSION_SELL_ORDERS	Общее число ордеров на продажу в текущий момент	long
SYMBOL_VOLUME	Volume - объем в последней сделке	long
SYMBOL_VOLUMEHIGH	Максимальный Volume за день	long
SYMBOL_VOLUMELOW	Минимальный Volume за день	long

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
SYMBOL_TIME	Время последней котировки	datetime
SYMBOL_TIME_MSC	Время последней котировки в миллисекундах с 1970.01.01	long
SYMBOL_DIGITS	Количество знаков после запятой	int
SYMBOL_SPREAD	Размер спреда в пунктах	int
SYMBOL_SPREAD_FLOAT	Признак плавающего спреда	bool
SYMBOL_TICKS_BOOKDEPTH	Максимальное количество показываемых заявок в стакане. Для инструментов, не имеющих очереди заявок, значение равно 0	int
SYMBOL_TRADE_CALC_MODE	Способ вычисления стоимости контракта	ENUM_SYMBOL_CALC_MODE
SYMBOL_TRADE_MODE	Тип исполнения ордеров	ENUM_SYMBOL_TRADE_MODE
SYMBOL_START_TIME	Дата начала торгов по инструменту (обычно используется для фьючерсов)	datetime
SYMBOL_EXPIRATION_TIME	Дата окончания торгов по инструменту (обычно используется для фьючерсов)	datetime
SYMBOL_TRADE_STOPS_LEVEL	Минимальный отступ в пунктах от текущей цены закрытия для установки Stop ордеров	int
SYMBOL_TRADE_FREEZE_LEVEL	Дистанция заморозки торговых операций (в пунктах)	int
SYMBOL_TRADE_EXEMODE	Режим заключения сделок	ENUM_SYMBOL_TRADE_EXECUTION

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
SYMBOL_SWAP_MODE	Модель расчета свопа	ENUM_SYMBOL_SWAP_MODE
SYMBOL_SWAP_ROLLOVER3DAYS	День недели для начисления тройного свопа	ENUM_DAY_OF_WEEK
SYMBOL_MARGIN_HEDGED_USE_LEG	Режим расчета хеджированной маржи по наибольшей стороне (Buy или Sell)	bool
SYMBOL_EXPIRATION_MODE	Флаги разрешенных режимов истечения ордера	int
SYMBOL_FILLING_MODE	Флаги разрешенных режимов заливки ордера	int
SYMBOL_ORDER_MODE	Флаги разрешенных типов ордера	int
SYMBOL_ORDER_GTC_MODE	Срок действия StopLoss и TakeProfit ордеров, если SYMBOL_EXPIRATION_MODE=SYMBOL_EXPIRATION_GTC (Good till cancelled)	ENUM_SYMBOL_ORDER_GTC_MODE
SYMBOL_OPTION_MODE	Тип опциона	ENUM_SYMBOL_OPTION_MODE
SYMBOL_OPTION_RIGHT	Право опциона (Call/Put)	ENUM_SYMBOL_OPTION_RIGHT

Для функции SymbolInfoDouble()

ENUM_SYMBOL_INFO_DOUBLE

Идентификатор	Описание	Тип свойства
SYMBOL_BID	Bid - лучшее предложение на продажу	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
SYMBOL_BIDHIGH	Максимальный Bid за день	double
SYMBOL_BIDLOW	Минимальный Bid за день	double
SYMBOL_ASK	Ask - лучшее предложение на покупку	double
SYMBOL_ASKHIGH	Максимальный Ask за день	double
SYMBOL_ASKLOW	Минимальный Ask за день	double
SYMBOL_LAST	Цена, по которой совершена последняя сделка	double
SYMBOL_LASTHIGH	Максимальный Last за день	double
SYMBOL_LASTLOW	Минимальный Last за день	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
SYMBOL_VOLUME_REAL	Объем последней совершенной сделки	double
SYMBOL_VOLUMEHIGH_REAL	Максимальный Volume за день	double
SYMBOL_VOLUMELOW_REAL	Минимальный Volume за день	double
SYMBOL_OPTION_STRIKE	Цена исполнения опциона. Это цена, по которой покупатель опциона может купить (при опционе Call) или продать (при опционе Put) базовый актив, а продавец	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	опциона соответственно обязан продать или купить соответствующее количество базового актива.	
SYMBOL_POINT	Значение одного пункта	double
SYMBOL_TRADE_TICK_VALUE	Значение SYMBOL_TRADE_TICK_VALUE_PROFIT	double
SYMBOL_TRADE_TICK_VALUE_PROFIT	Рассчитанная стоимость тика для прибыльной позиции	double
SYMBOL_TRADE_TICK_VALUE_LOSS	Рассчитанная стоимость тика для	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	убыточной позиции	
SYMBOL_TRADE_TICK_SIZE	Минимальное изменение цены	double
SYMBOL_TRADE_CONTRACT_SIZE	Размер торгового контракта	double
SYMBOL_TRADE_ACCRUED_INTEREST	Накопленный купонный доход - часть купонного процентного дохода по облигации, рассчитываемая пропорционально количеству дней, прошедших от даты выпуска купонной облига	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ции или даты выплаты предшествующего купонного дохода	
SYMBOL_TRADE_FACE_VALUE	Номинальная стоимость - начальная стоимость облигации, установленная эмитентом	double
SYMBOL_TRADE_LIQUIDITY_RATE	Коэффициент ликвидности - доля от стоимости актива, которую можно использовать в качестве залога.	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
SYMBOL_VOLUME_MIN	Минимальный объем для заключения сделки	double
SYMBOL_VOLUME_MAX	Максимальный объем для заключения сделки	double
SYMBOL_VOLUME_STEP	Минимальный шаг изменения объема для заключения сделки	double
SYMBOL_VOLUME_LIMIT	Максимально допустимый для данного символа совокупный объем открытой позиции и отложенных ордеров в одном	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	направлении (покупка или продажа). Например, при ограничении в 5 лотов можно иметь открытую позицию на покупку объемом 5 лотов и выставить отложенный ордер Sell Limit объемом 5 лотов. Но при этом нельзя выставить отложенный ордер Buy Limit (поскольку совоку	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	пный объем в одном направлении превысит ограничение) или выставить Sell Limit объемом более 5 лотов.	
SYMBOL_SWAP_LONG	Значение свопа в покупку	double
SYMBOL_SWAP_SHORT	Значение свопа в продажу	double
SYMBOL_SWAP_SUNDAY	Коэффициент начисления свопов (SYMBOL_SWAP_LONG или SYMBOL_SWAP_SHORT) при перено	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	се позиции с указанного дня (SUNDAY) на следующий. Могут быть следующие значения: • 0 - свопы не начисляются • 1 - одиночное	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	а ч и с л е н и е с в о п о в • 3 - т р о й н о е н а ч и с л е н и е с в о п о в	
SYMBOL_SWAP_MONDAY	Коэффициент начисления свопов (SYMB	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	OL_SWAP_LONG или SYMBOL_SWAP_SHORT) при переносе позиции с понедельника на вторник	
SYMBOL_SWAP_TUESDAY	Коэффициент начисления свопов (SYMBOL_SWAP_LONG или SYMBOL_SWAP_SHORT) при переносе позиции со вторника на среду	double
SYMBOL_SWAP_WEDNESDAY	Коэффициент начисления свопов (SYMBOL_SWAP_LONG или	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	SYMBOL_SWAP_SHORT) при переносе позиции со среды на четверг	
SYMBOL_SWAP_THURSDAY	Коэффициент начисления свопов (SYMBOL_SWAP_LONG или SYMBOL_SWAP_SHORT) при переносе позиции с четверга на пятницу	double
SYMBOL_SWAP_FRIDAY	Коэффициент начисления свопов (SYMBOL_SWAP_LONG или SYMBOL_SWAP_SHO	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	RT) при переносе позиции с пятницы на субботу	
SYMBOL_SWAP_SATURDAY	Коэффициент начисления свопов (SYMBOL_SWAP_LONG или SYMBOL_SWAP_SHORT) при переносе позиции с субботы на воскресенье	double
SYMBOL_MARGIN_INITIAL	Начальная (инициирующая) маржа обозначает размер необходимых залоговых средств в	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	маржинальной валюте для открытия позиции объемом в один лот. Используется при проверке средств клиента при входе в рынок. Для получения информации о размере взимаемой маржи в зависимости от типа и направления ордера используется	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	функция SymbolInfoMarginRate().	
SYMBOL_MARGIN_MAINTENANCE	Поддерживающая маржа по инструменту. В случае если задана - указывает размер маржи в маржинальной валюте инструмента, удерживаемой с одного лота. Используется при проверке средств клиента при изменении состояния счета	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	клиента. Если поддерживающая маржа равна 0, то используется начальная маржа. Для получения информации о размере взимаемой маржи в зависимости от типа и направления ордера используется функция SymbolInfoMarginRate().	
SYMBOL_SESSION_VOLUME	Суммарный объём сделок в текущу	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ю сессию	
SYMBOL_SESSION_TURNOVER	Суммарный оборот в текущую сессию	double
SYMBOL_SESSION_INTEREST	Суммарный объём открытых позиций	double
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Общий объём ордеров на покупку в текущий момент	double
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Общий объём ордеров на продажу в текущий момент	double
SYMBOL_SESSION_OPEN	Цена открытия сессии	double
SYMBOL_SESSION_CLOSE	Цена закрытия сессии	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
SYMBOL_SESSION_AW	Средневзвешенная цена сессии	double
SYMBOL_SESSION_PRICE_SETTLEMENT	Цена поставки на текущую сессию	double
SYMBOL_SESSION_PRICE_LIMIT_MIN	Минимально допустимое значение цены на сессию	double
SYMBOL_SESSION_PRICE_LIMIT_MAX	Максимально допустимое значение цены на сессию	double
SYMBOL_MARGIN_HEDGED	Размер контракта или маржи для одного лота перекрытых позиций (разнонаправленные позици	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	и по одному символу). Существует два способа расчета маржи для перекрытых позиций. Способ расчета определяется брокером. Базовый расчет: • Если для инструмента задана первоначальная маржа (SYMBOL_MARGIN_INITIAL),	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	то хеджированная маржа указывается как абсолютное значение (в деньгах). • Если первоначальная маржа не задана (равна 0), то в SYMBOL_MARGIN_HEDGED указывается размер контракта, кото	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	рый будет использован при расчете маржи по формуле, соответствующей типу торгового инструмента (SYMBOL_TRADE_CALC_MODE). Расчет по наибольшей позиции: • Значение SYMBOL_MARGIN_HEDGED	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	не учитывается. • Вычисляется объем всех коротких и всех длинных позиций по инструменту. • Для каждой стороны рассчитывается средневзвешенная цена открытия, а также средневзвешенная цена конверта	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ции в валюту депозита. • Далее по формулам, соответствующим типу инструмента (SYMBOL_TRADE_CALC_MODE), рассчитывается маржа для короткой и для длинной стороны. • В качестве итогового знач	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ения использ уется наиб ольш ее.	
SYMBOL_PRICE_CHANGE	Измен ение текуще й цены относи тельно конца преды дущего торгов ого дня, выраж енное в проце нтах	double
SYMBOL_PRICE_VOLATILITY	Волати льност ь цены в проце нтах	double
SYMBOL_PRICE_THEORETICAL	Теорет ическа я цена опцион а	double
SYMBOL_PRICE_DELTA	Дельта опцион а/варр анта. Показы вает, на скольк о единиц измени	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	тся цена опциона при изменении цены базового актива на 1 единицу	
SYMBOL_PRICE_THETA	Тета опциона/варранта. Количество пунктов, которое будет терять цена опциона каждый день из-за временного распада, т.е. при приближении даты экспирации	double
SYMBOL_PRICE_GAMMA	Гамма опциона/варранта. Показывает	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	скорость изменения дельты - насколько быстро или медленно меняется опционная премия	
SYMBOL_PRICE_VEGA	Вега опциона/варранта. Показывает количество пунктов, на которое изменится цена опциона при изменении волатильности на 1%	double
SYMBOL_PRICE_RHO	Ро опциона/варранта. Отражает чувствительн	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ость теоретической цены опциона к изменению процентной ставки на 1%	
SYMBOL_PRICE_OMEGA	Омега опциона/варранта. Эластичность опциона - относительное процентное изменение цены опциона на процентное изменение цены базового актива	double
SYMBOL_PRICE_SENSITIVITY	Чувствительность опциона/варранта. Показывает, на	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	сколько пунктов должна измениться цена базового актива опциона, чтобы цена опциона изменилась на один пункт	

Для функции SymbolInfoString()

ENUM_SYMBOL_INFO_STRING

Идентификатор	Описание	Тип свойства
SYMBOL_BASIS	Имя базового актива для производного инструмента	string
SYMBOL_CATEGORY	Название категории или сектора, к которой принадле	string

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	жит финансовый инструмент	
SYMBOL_COUNTRY	Страна, к которой отнесен финансовый инструмент	string
SYMBOL_SECTOR_NAME	Сектор экономики, к которому относится финансовый инструмент	string
SYMBOL_INDUSTRY_NAME	Отрасль экономики или вид промышленности, к которой относится финансовый инструмент	string
SYMBOL_CURRENCY_BASE	Базовая валюта инструмента	string
SYMBOL_CURRENCY_PROFIT	Валюта прибыли	string

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
SYMBOL_CURRENCY_MARGIN	Валюта, в которой вычисляется залоговые средства	string
SYMBOL_BANK	Источник текущей котировки	string
SYMBOL_DESCRIPTION	Строковое описание символа	string
SYMBOL_EXCHANGE	Название биржи или торговой площадки, на которой торгуется символ	string
SYMBOL_FORMULA	Формула для построения цены пользовательского символа. Если имя финансового инструмента, входящего в формулу, начинается с цифры или содержи	string

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	т спецсимвол (" ", ".", "-", "&", "#" и т.д.), то имя этого инструмента должно быть заключено в кавычки. • Синтетический символ: "@ESU19" / EUR	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	CAD • Календарный спред: "Si-9.13"-"Si-6.13" • Индексев р	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	o:34.388057266*pow(EURUSD,0.3155)*pow(EURGBP,0.30	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	56)*pow(EURJPY,0.18911)*pow(EURCHF,0.11113)*pow(EU	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	RSEK, 0.07855)	
SYMBOL_ISIN	Имя торгового символа в системе международных идентификационных кодов ценных бумаг — ISIN (International Securities Identification Number). Международный идентификационный код ценной бумаги — это 12-разрядный буквенно-цифровой код,	string

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	однозначно идентифицирующий ценную бумагу. Наличие данного свойства символа определяется на стороне торгового сервера.	
SYMBOL_PAGE	Адрес интернет страницы с информацией по символу. Данный адрес будет отображаться в виде ссылки при просмотре свойств символа в терминале	string
SYMBOL_PATH	Путь в дереве символов	string

Ценовой график по символу может строиться на основе цены Bid или Last. От выбора цены для построения графиков зависит то, как формируются и отображаются бары в терминале. Возможные значения свойства SYMBOL_CHART_MODE приводятся в перечислении ENUM_SYMBOL_CHART_MODE

© 2000-2025, MetaQuotes Ltd.

ENUM_SYMBOL_CHART_MODE

Идентификатор	Описание
SYMBOL_CHART_MODE_BID	Бары строятся по ценам Bid
SYMBOL_CHART_MODE_LAST	Бары строятся по ценам Last

Для каждого финансового инструмента могут быть указаны несколько режимов срока действия (истечения) отложенных ордеров. Каждому режиму сопоставлен флаг, флаги могут комбинироваться операцией логического **ИЛИ** (|), например, SYMBOL_EXPIRATION_GTC|SYMBOL_EXPIRATION_SPECIFIED. Чтобы проверить разрешенность конкретного режима для инструмента, необходимо результат логического **И** (&) сравнить с флагом режима.

Если для символа указан флаг SYMBOL_EXPIRATION_SPECIFIED, то при отправке отложенного ордера можно конкретно указать, до какого момента действует данный отложенный ордер.

Идентификатор	Значение	Описание
SYMBOL_EXPIRATION_GTC	1	Ордер действителен неограниченно по времени до явной его отмены
SYMBOL_EXPIRATION_DAY	2	Ордер действителен до конца дня
SYMBOL_EXPIRATION_SPECIFIED	4	Срок истечения указывается в ордере
SYMBOL_EXPIRATION_SPECIFIED_DAY	8	День истечения указывается в ордере

Пример:

```
//+------------------------------------------------------------------+
//| проверяет разрешенность указанного режима экспирации             |
//+------------------------------------------------------------------+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
  {
//--- получим значение свойства, описывающего допустимые режимы истечения срока дейст
   int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- вернем true, если режим exp_type разрешен
   return((expiration&exp_type)==exp_type);
  }
```

Если свойство SYMBOL_EXPIRATION_MODE имеет значение SYMBOL_EXPIRATION_GTC (ордер действителен до отмены), то срок действия отложенных ордеров и установленных уровней StopLoss/TakeProfit дополнительно задается с помощью перечисления ENUM_SYMBOL_ORDER_GTC_MODE.

© 2000-2025, MetaQuotes Ltd.

ENUM_SYMBOL_ORDER_GTC_MODE

Идентификатор	Описание
SYMBOL_ORDERS_GTC	Отложенные ордеры и уровни Stop Loss/Take Profit действительны неограниченно по времени до явной отмены
SYMBOL_ORDERS_DAILY	Ордеры действуют только внутри одного торгового дня. По его завершении все уровни StopLoss и TakeProfit удаляются, а также удаляются отложенные ордеры
SYMBOL_ORDERS_DAILY_EXCLUDING_STOPS	При смене торгового дня удаляются только отложенные ордеры, уровни StopLoss и TakeProfit сохраняются

При отправке ордера можно указать политику заполнения заявленного в торговом приказе объема. Допустимые варианты исполнения ордера по объему для каждого символа указаны в таблице. Для каждого инструмента может быть установлен не один режим, а несколько через комбинацию флагов. Комбинация флагов выражается операцией логического **ИЛИ** (|), например, SYMBOL_FILLING_FOK|SYMBOL_FILLING_IOC. Чтобы проверить разрешенность конкретного режима для инструмента, необходимо результат логического **И** (&) сравнить с флагом режима – пример.

Политика заполнения	Идентификатор	Значение	Описание
Все/Ничего	SYMBOL_FILLING_FOK	1	Ордер может быть исполнен

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			лнени с ключительно в указанном объеме. Еслина

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			рынке в данный момент не присутствует достаточно

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			гообъемафинансовогоинструмента, тоордернебу

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			детиспользуется. Необходимый объем может быть больше

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			вленизнескольких предложений, доступных в данн

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			ыймоментнарынке.Дляданнойполитикиприотпра

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			в кеордеране обхомоуказывать типзаливки ORDER_

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			FILLING_FOK. Возможность использования FOK-о

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			рдеровопределяетсянаторговомсервере.
Все/Частично	SYMBOL_FILLING_IOC	2	Трейде

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			рсоглашаетсясоверршитьсделкуупомаксималь нодо

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			ступномунарынкеобъемувпределахуказанногово

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			рдере. Вслучаеневозможностиполногоисполнени

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			я ордер будет исполнен на доступный объем, а не исп

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			олненный объем ордера будет отменен. Для данно

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			йполитикиприотправкеордераненеобходимоуказывать

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			ьтипзаливкиORDER_FILLING_IOC. Возможности

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			спользования IOC-ордеровопределяется на торго

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			в ордере.
Пассивная	SYMBOL_FILLING_BOC	4	Политика BOC (Book-or-Cancel) пред

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			полагает, что ордер может быть только выставлен в

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			стаканцени не может быть немедленно исполнен.

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			Если ордер может быть исполнен немедленно при высоких

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			тавлении, тоонснимается. Фактически, данную п

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			политики использования можно указывать только вместе

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			случае, когда цена выставленного ордера будет ближе

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			етекущегорынка.ВоС-ордерыиспользуютсядляре

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			ализациипассивнойторговли,чтобыордерагарант

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			ированно не были исполнены сразу при выставлении ин

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			еповлиялнатекущуюликвидность. Поддерживае

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			тся только для лимитных истоп-лимитных ордеров,

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			тоесть, еслифлаг SYMBOL_ORDER_MODE содержитзн

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			ачения SYMBOL_ORDER_LIMITи/или SYMBOL_ORDER_

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			STOP_LIMIT.
Вернуть	Идентификатор отсутствует		В случае частичного исполнения ярыно

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			чный или лимитный ордер с остаточным объемом не сн

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			имается, а продолжает действовать. Для данной п

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			политики приоритправкеордераненеобходимоуказыватьт

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			инзаливкиORDER_FILLING_RETURN. Return-орде

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			рынке разрешены в режиме исполнения Market Execut

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			ion(исполнениеордеровпорынку_SYMBOL_TRADE_

© 2000-2025, MetaQuotes Ltd.

Политика заполнения	Идентификатор	Значение	Описание
			EXECUTION_MARKET).

При отправке торгового запроса функцией OrderSend() нужную политику исполнения по объему можно задать в поле *type_filling* в специальной структуре MqlTradeRequest, допустимы значения из перечисления ENUM_ORDER_TYPE_FILLING. Если тип заливки не указан, то в торговом запросе автоматически подставится ORDER_FILLING_RETURN. При этом нужно отметить, что тип заливки ORDER_FILLING_RETURN разрешен при любых режимах исполнения, кроме режима "Исполнение по рынку" (SYMBOL_TRADE_EXECUTION_MARKET).

При отправке торгового запроса на исполнение **в действующий момент времени** (time in force) необходимо учитывать, что на финансовых рынках нет гарантии того, что в данный момент времени по данному финансовому инструменту доступен весь запрашиваемый объем по желаемой цене. Поэтому проведение торговых операций в реальном времени регулируется с помощью режимов исполнения по цене и объему. Режимы, или по-другому политики исполнения, определяют правила для случаев, когда изменилась цена или запрашиваемый объем нельзя выполнить полностью **в текущий момент времени**.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
Режим исполнения (Request Execution)	Исполнение рыночного ордера по	SYMBOL_TRADE_EXECUTION_REQUEST

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	цене, предварительно полученной от брокера. Перед отправкой рыночного ордера у брокера запрашиваются цены его исполнения. После их получения выполнение ордера по данной цене можно либо подтвердить, либо отклонить.	
Немедленное исполнение (Instant Execution)	Немедленное исполнение рыночного	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	ордера по указанной цене. При отправке торгового запроса на исполнение платформа автоматически подставляет в ордер текущие цены. • Если брокер принимает ордер	

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	ну, то ордер будет исполнен. • Если брокер не принимает запр	

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	рошенную цену, то про исходит так называемое "реквотиро	

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	вание "(Requote)_брокер возвращает цены, по которым	

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	жетбытьисполненданныйордер.	
Исполнение по рынку (Market Execution)	Решение оцене исполнения принимает брокер без дополнительного согласования с трейдером. Отправка	SYMBOL_TRADE_EXECUTION_MARKET

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	рыночного ордера в таком режиме подразумевает досрочное согласие с ценой, по которой он будет выполнен.	
Биржевое исполнение (Exchange Execution)	Торговые операции совершаются по ценам текущих рыночных предложений.	SYMBOL_TRADE_EXECUTION_EXCHANGE

Перед отправкой ордера с исполнением в текущий момент времени, для правильного выставления значения ORDER_TYPE_FILLING (тип исполнения по объему), для каждого финансового инструмента с помощью функции SymbolInfoInteger() можно получить значение свойства SYMBOL_FILLING_MODE, которое показывает разрешенные для этого символа типы заполнения по объему в виде комбинации флагов. При этом нужно отметить, что тип заливки ORDER_FILLING_RETURN разрешен всегда, кроме режима "Исполнение по рынку" (SYMBOL_TRADE_EXECUTION_MARKET).

Применение типов заливки в зависимости от режима исполнения можно представить в виде таблицы:

© 2000-2025, MetaQuotes Ltd.

Режим исполнения\Политика заполнения	Все/Ничего (FOK ORDER_FILLING_FOK)	Все/Частично (IOC ORDER_FILLING_IOC)	Вернуть (Return ORDER_FILLING_RETURN)
Немедленное исполнение (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (независимо от настройки символа)	+ (независимо от настройки символа)	+ (всегда)
Исполнения по запросу SYMBOL_TRADE_EXECUTION_REQUEST	+ (независимо от настройки символа)	+ (независимо от настройки символа)	+ (всегда)
Исполнение по рынку SYMBOL_TRADE_EXECUTION_MARKET	+ (задается в настройках символа)	+ (задается в настройках символа)	- (запрещено при любых настройках символа)
Биржевое исполнение SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (задается в настройках символа)	+ (задается в настройках символа)	+ (всегда)

Для отложенных ордеров независимо от режима исполнения (SYMBOL_TRADE_EXEMODE) необходимо использовать тип заливки ORDER_FILLING_RETURN, так как такие ордеры не предназначены для исполнения в момент отправки. При использовании отложенных ордеров трейдер заранее соглашается, что при наступлении условий для проведения сделки по данному ордеру, брокером будет использован тот тип заливки, который поддерживается данной торговой площадкой.

Пример:

```
//+------------------------------------------------------------------+
//| проверяет разрешенность указанного режима заполнения             |
//+------------------------------------------------------------------+
bool IsFillingTypeAllowed(string symbol,int fill_type)
  {
//--- получим значение свойства, описывающего режим заполнения
   int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- вернем true, если режим fill_type разрешен
   return((filling&fill_type)==fill_type);
  }
```

При отправке торгового запроса функцией OrderSend() для некоторых операций необходимо указать тип ордера из перечисления ENUM_ORDER_TYPE. Не все типы ордеров могут быть разрешены для конкретного финансового инструмента, свойство SYMBOL_ORDER_MODE описывает флаги разрешенных типов ордеров.

Идентификатор	Значение	Описание
SYMBOL_ORDER_MARKET	1	Разрешены рыночные ордера

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение	Описание
		(Buy и Sell)
SYMBOL_ORDER_LIMIT	2	Разрешены лимитные ордера (Buy Limit и Sell Limit)
SYMBOL_ORDER_STOP	4	Разрешены стоп-ордера (Buy Stop и Sell Stop)
SYMBOL_ORDER_STOP_LIMIT	8	Разрешены стоп-лимит ордера (Buy Stop Limit и Sell Stop Limit)
SYMBOL_ORDER_SL	16	Разрешена установка Stop Loss
SYMBOL_ORDER_TP	32	Разрешена установка Take Profit
SYMBOL_ORDER_CLOSEBY	64	Разрешение на закрытие позиции с помощью встречной - то есть открытой в противоположном направлении по тому же инструменту. Свойство задается для счетов с хеджинговой системой учета (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)

Пример:

```
//+------------------------------------------------------------------+
//| Функция выводит на печать разрешенные для символа типы ордеров   |
//+------------------------------------------------------------------+
void Check_SYMBOL_ORDER_MODE(string symbol)
  {
//--- получим значение свойства, описывающего разрешенные типы ордеров
   int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
//--- проверка на рыночные ордера (Market Execution)
   if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
      Print(symbol+": Рыночные ордера разрешены (Buy и Sell)");
//--- проверка на ордера типа Limit
   if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
      Print(symbol+": Ордера Buy Limit и Sell Limit разрешены");
//--- проверка на ордера типа Stop
   if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
      Print(symbol+": Ордера Buy Stop и Sell Stop разрешены");
//--- проверка на ордера типа Stop Limit
   if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
      Print(symbol+": Ордера Buy Stop Limit и Sell Stop Limit разрешены");
//--- проверка на возможность установки ордеров Stop Loss
   if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
      Print(symbol+": Ордера Stop Loss разрешены");
//--- проверка на возможность установки ордеров Take Profit
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(symbol+": Ордера Take Profit разрешены");
//--- проверка разрешения на закрытие позиции с помощью встречной
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_CLOSEBY)
      Print(symbol+": Встречное закрытие разрешено");
//---
  }
```

Для получения информации о способе вычисления величины залоговых средств по инструменту (размера маржинальных требований) предназначено перечисление ENUM_SYMBOL_CALC_MODE.

ENUM_SYMBOL_CALC_MODE

Идентификатор	Описание	Формула
SYMBOL_CALC_MODE_FOREX	Forex mode - расчет прибыли и маржи для Форекс	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 - расчет прибыли и маржи для Форекс без учета	Margin: Lots * Contract_Size * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	плеча	
SYMBOL_CALC_MODE_FUTURES	Futures mode - расчет залога и прибыли для фьючерсов	Margin: Lots * InitialMargin * Margin_Rate Profit: (close_price - open_price) * TickPrice / TickSize*Lots
SYMBOL_CALC_MODE_CFD	CFD mode - расчет залога и прибыли для CFD	Margin: Lots * ContractSize * MarketPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_CFDINDEX	CFD index mode - расчет залога и прибыли для CFD на индексы	Margin: (Lots * ContractSize * MarketPrice) * TickPrice / TickSize * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD Leverage	Margin: (Lots * ContractSize * MarketPrice) / Leverage * Margin_Rate

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	mode - расчет залога и прибыли для CFD при торговле с плечом	Profit: (close_price-open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_STOCKS	Exchange mode - расчет залога и прибыли для торговли ценными бумагами на бирже	Margin: Lots * ContractSize * LastPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_FUTURES	Futures mode - расчет залога и прибыли для	Margin: Lots * InitialMargin * Margin_Rate или Lots * MaintenanceMargin * Margin_Rate Profit: (close_price - open_price) * Lots * TickPrice / TickSize

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	торговлифьючерсными контрактами на бирже	
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS Futures mode - расчет залога и прибыли для торговлифьючерсными контрактами на FORTS. Размер маржи может уменьшаться на вели	Margin: Lots * InitialMargin * Margin_Rate или Lots * MaintenanceMargin * Margin_Rate Profit: (close_price - open_price) * Lots * TickPrice / TickSize

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	чину отклонения MarginDiscount по следующим правилам: 1. Если цена длинной позиции (ордера на покупку) меньше расчетной цены, то MarginDiscount = Lots*((PriceSettle-PriceOrder)*TickPrice/Tic	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	kSize) 2. Если цена короткой позиций (ордера на продажу) больше расчетной цены, то MarginDiscount = Lots*((PriceOrder-PriceSettle)*TickPrice/TickSize) где: ☐ PriceSettl	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	е - расчетная (клиринговая) цена предыдущей сессии; ☐ Pri	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	средневзвешенная цена позиции или цена отложенного ордера	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	ия, указанная в ордере (заявке); ☐TickPrice-ценатика (с	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	тоимостьизменениянаодинпункт); ☐TickSize - размере	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	ртика (минимальный шаг изменения цены)	
SYMBOL_CALC_MODE_EXCH_BONDS	Exchange Bonds mode - расчет залога и прибыли для торг	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	овли облигациями на бирже	
SYMBOL_CALC_MODE_EXCH_STOCKS_MOEX	Exchange MOEX Stocks mode - расчет залога и прибыли для торговли ценными бумагами на бирже MOEX	Margin: Lots * ContractSize * LastPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_BONDS_MOEX	Exchange MOEX Bonds mode - расчет залога и прибыли	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	для торговли облигациями на бирже MOEX	
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - инструмент используется в качестве неторгуемого актива на торговом счете. Рыночная стоимость открытой позиции рассчитывается на осно	Margin: нет Profit: нет Рыночная стоимость: Lots * ContractSize * MarketPrice * LiqudityRate

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	вании объема, текущей цены рынка, размера контракта и коэффициента ликвидности. Стоимость учитывается в Активах (Assets), которые суммируются с собственными средствами (Equity). Тем самым откр	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Формула
	ытые позиции по такому инструменту увеличивают размер свободных средств (Free Margin) и служат дополнительным обеспечением под открытые позиции по торгуемым инструментам	

Существует несколько режимов торговли по финансовым инструментам. Информация о режимах торговли по конкретному инструменту отображена в значениях перечисления ENUM_SYMBOL_TRADE_MODE.

© 2000-2025, MetaQuotes Ltd.

ENUM_SYMBOL_TRADE_MODE

Идентификатор	Описание
SYMBOL_TRADE_MODE_DISABLED	Торговля по символу запрещена
SYMBOL_TRADE_MODE_LONGONLY	Разрешены только покупки
SYMBOL_TRADE_MODE_SHORTONLY	Разрешены только продажи
SYMBOL_TRADE_MODE_CLOSEONLY	Разрешены только операции закрытия позиций
SYMBOL_TRADE_MODE_FULL	Нет ограничений на торговые операции

Возможные режимы заключения сделок по конкретному инструменту определены в перечислении ENUM_SYMBOL_TRADE_EXECUTION.

ENUM_SYMBOL_TRADE_EXECUTION

Идентификатор	Описание
SYMBOL_TRADE_EXECUTION_REQUEST	Торговля по запросу
SYMBOL_TRADE_EXECUTION_INSTANT	Торговля по потоковым ценам
SYMBOL_TRADE_EXECUTION_MARKET	Исполнение ордеров по рынку
SYMBOL_TRADE_EXECUTION_EXCHANGE	Биржевое исполнение

Способы начисления свопов при переносе позиции указаны в перечислении ENUM_SYMBOL_SWAP_MODE. Способ начисления свопов определяет единицы измерения параметров SYMBOL_SWAP_LONG и SYMBOL_SWAP_SHORT. Например, если свопы начисляются в валюте депозита клиента, в параметрах объем начисляемых свопов указывается именно в валюте депозита клиента.

ENUM_SYMBOL_SWAP_MODE

Идентификатор	Описание
SYMBOL_SWAP_MODE_DISABLED	Нет свопов
SYMBOL_SWAP_MODE_POINTS	Свопы начисляются в пунктах
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Свопы начисляются в деньгах в базовой валюте символа
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Свопы начисляются в деньгах в маржинальной валюте символа
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Свопы начисляются в деньгах в валюте депозита клиента
SYMBOL_SWAP_MODE_CURRENCY_PROFIT	Свопы начисляются в деньгах в валюте расчета прибыли

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Свопы начисляются в годовых процентах от цены инструмента на момент расчета свопа(банковский режим – 360 дней в году)
SYMBOL_SWAP_MODE_INTEREST_OPEN	Свопы начисляются в годовых процентах от цены открытия позиции по символу (банковский режим – 360 дней в году)
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Свопы начисляются переоткрытием позиции. В конце торгового дня позиция принудительно закрывается. На следующий день позиция переоткрывается по цене закрытия +/- указанное количество пунктов (в параметрах SYMBOL_SWAP_LONG и SYMBOL_SWAP_SHORT)
SYMBOL_SWAP_MODE_REOPEN_BID	Свопы начисляются переоткрытием позиции. В конце торгового дня позиция принудительно закрывается. На следующий день позиция переоткрывается по текущей цене Bid +/- указанное количество пунктов (в параметрах SYMBOL_SWAP_LONG и SYMBOL_SWAP_SHORT)

Для указания дня недели предназначены значения перечисления ENUM_DAY_OF_WEEK.

ENUM_DAY_OF_WEEK

Идентификатор	Описание
SUNDAY	Воскресенье
MONDAY	Понедельник
TUESDAY	Вторник
WEDNESDAY	Среда
THURSDAY	Четверг
FRIDAY	Пятница
SATURDAY	Суббота

Опцион – это контракт, который дает право, но не обязанность купить или продать базовый актив (товар, акцию, фьючерс и т.д.) по фиксированной цене в течении жизни опциона или в определенный момент времени. Для описания свойств опционов предназначены перечисления, которые описывают тип опциона и право, который он представляет.

ENUM_SYMBOL_OPTION_RIGHT

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
SYMBOL_OPTION_RIGHT_CALL	Опцион, дающий право купить актив по фиксированной цене
SYMBOL_OPTION_RIGHT_PUT	Опцион, дающий право продать актив по фиксированной цене

ENUM_SYMBOL_OPTION_MODE

Идентификатор	Описание
SYMBOL_OPTION_MODE_EUROPEAN	Европейский тип опциона - может быть погашен только в указанную дату (дата истечения срока, дата исполнения, дата погашения)
SYMBOL_OPTION_MODE_AMERICAN	Американский тип опциона - может быть погашен в любой день до истечения срока опциона. Для такого типа задаётся период, в течение которого покупатель может исполнить данный опцион

Финансовые инструменты распределяются по секторам экономики. Сектор экономики определяется общими характеристиками, экономическими целями, функциями и поведением, что позволяет отделить данный сектор от других частей экономики. В ENUM_SYMBOL_SECTOR перечислены секторы экономики, к которым может относиться торговый инструмент.

ENUM_SYMBOL_SECTOR

Идентификатор	Описание
SECTOR_UNDEFINED	Не определен
SECTOR_BASIC_MATERIALS	Сырье
SECTOR_COMMUNICATION_SERVICES	Услуги связи
SECTOR_CONSUMER_CYCLICAL	Потребление циклического спроса
SECTOR_CONSUMER_DEFENSIVE	Основное потребление
SECTOR_CURRENCY	Валюты
SECTOR_CURRENCY_CRYPTO	Криптовалюты
SECTOR_ENERGY	Энергетика
SECTOR_FINANCIAL	Финансы
SECTOR_HEALTHCARE	Здравоохранение
SECTOR_INDUSTRIALS	Промышленность
SECTOR_REAL_ESTATE	Недвижимость

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
SECTOR_TECHNOLOGY	Технологии
SECTOR_UTILITIES	Коммунальные услуги
SECTOR_INDEXES	Индексы
SECTOR_COMMODITIES	Биржевые товары

Каждый финансовый инструмент может быть отнесен к определенному виду промышленности или отрасли. Отрасль экономики — совокупность предприятий, производящих (добывающих) однородную или специфическую продукцию по однотипным технологиям. В ENUM_SYMBOL_INDUSTRY перечислены виды промышленности, к которым может относиться торговый инструмент.

ENUM_SYMBOL_INDUSTRY

Идентификатор	Описание
INDUSTRY_UNDEFINED	Не определено
Сырье	
INDUSTRY_AGRICULTURAL_INPUTS	Сельскохозяйственные ресурсы
INDUSTRY_ALUMINIUM	Алюминий
INDUSTRY_BUILDING_MATERIALS	Строительные материалы
INDUSTRY_CHEMICALS	Химикаты
INDUSTRY_COKING_COAL	Коксующийся уголь
INDUSTRY_COPPER	Медь
INDUSTRY_GOLD	Золото
INDUSTRY_LUMBER_WOOD	Производство пиломатериалов и древесины
INDUSTRY_INDUSTRIAL_METALS	Прочие промышленные металлы и добыча
INDUSTRY_PRECIOUS_METALS	Прочие драгоценные металлы и добыча
INDUSTRY_PAPER	Целлюлозно-бумажные изделия
INDUSTRY_SILVER	Серебро
INDUSTRY_SPECIALTY_CHEMICALS	Специальные химикаты
INDUSTRY_STEEL	Сталь
Услуги связи	
INDUSTRY_ADVERTISING	Рекламные агентства

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
INDUSTRY_BROADCASTING	Вещание
INDUSTRY_GAMING_MULTIMEDIA	Электронные игры и мультимедиа
INDUSTRY_ENTERTAINMENT	Развлечения
INDUSTRY_INTERNET_CONTENT	Интернет-контент и информация
INDUSTRY_PUBLISHING	Издательство
INDUSTRY_TELECOM	Телекоммуникационные услуги
Потребление циклического спроса	
INDUSTRY_APPAREL_MANUFACTURING	Производство одежды
INDUSTRY_APPAREL_RETAIL	Розничная продажа одежды
INDUSTRY_AUTO_MANUFACTURERS	Автомобилестроение
INDUSTRY_AUTO_PARTS	Автозапчасти
INDUSTRY_AUTO_DEALERSHIP	Дилеры легковых и грузовых автомобилей
INDUSTRY_DEPARTMENT_STORES	Универсальные магазины
INDUSTRY_FOOTWEAR_ACCESSORIES	Обувь и аксессуары
INDUSTRY_FURNISHINGS	Мебель, фурнитура и бытовая техника
INDUSTRY_GAMBLING	Игорные предприятия
INDUSTRY_HOME_IMPROV_RETAIL	Розничная торговля товарами для дома
INDUSTRY_INTERNET_RETAIL	Розничная онлайн-торговля
INDUSTRY_LEISURE	Досуг
INDUSTRY_LODGING	Жилье
INDUSTRY_LUXURY_GOODS	Товары класса "люкс"
INDUSTRY_PACKAGING_CONTAINERS	Упаковка
INDUSTRY_PERSONAL_SERVICES	Персональные услуги
INDUSTRY_RECREATIONAL_VEHICLES	Транспортные средства для отдыха
INDUSTRY_RESIDENT_CONSTRUCTION	Жилищное строительство
INDUSTRY_RESORTS_CASINOS	Курорты и казино
INDUSTRY_RESTAURANTS	Рестораны
INDUSTRY_SPECIALTY_RETAIL	Специализированная розничная торговля
INDUSTRY_TEXTILE_MANUFACTURING	Текстильное производство
INDUSTRY_TRAVEL_SERVICES	Туристические услуги

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
Основное потребление	
INDUSTRY_BEVERAGES_BREWERS	Напитки - Пивовары
INDUSTRY_BEVERAGES_NON_ALCO	Напитки - Безалкогольные
INDUSTRY_BEVERAGES_WINERIES	Напитки - Винзаводы и ликеро-водочные заводы
INDUSTRY_CONFECTIONERS	Кондитеры
INDUSTRY_DISCOUNT_STORES	Дисконтные магазины
INDUSTRY_EDUCATION_TRAINIG	Образование и обучение
INDUSTRY_FARM_PRODUCTS	Сельскохозяйственные продукты
INDUSTRY_FOOD_DISTRIBUTION	Дистрибуция продуктов питания
INDUSTRY_GROCERY_STORES	Продуктовые магазины
INDUSTRY_HOUSEHOLD_PRODUCTS	Товары для дома и быта
INDUSTRY_PACKAGED_FOODS	Упакованные продукты
INDUSTRY_TOBACCO	Табак
Энергетика	
INDUSTRY_OIL_GAS_DRILLING	Бурение нефтяных и газовых скважин
INDUSTRY_OIL_GAS_EP	Добыча и переработка нефти и газа
INDUSTRY_OIL_GAS_EQUIPMENT	Нефтегазовое оборудование и услуги
INDUSTRY_OIL_GAS_INTEGRATED	Интегрированные нефтегазовые компании
INDUSTRY_OIL_GAS_MIDSTREAM	Транспортировка нефти и газа
INDUSTRY_OIL_GAS_REFINING	Переработка и сбыт нефти и газа
INDUSTRY_THERMAL_COAL	Энергетический уголь
INDUSTRY_URANIUM	Уран
Финансы	
INDUSTRY_EXCHANGE_TRADED_FUND	Биржевой фонд
INDUSTRY_ASSETS_MANAGEMENT	Управление активами
INDUSTRY_BANKS_DIVERSIFIED	Банки - Диверсифицированные
INDUSTRY_BANKS_REGIONAL	Банки - Региональные
INDUSTRY_CAPITAL_MARKETS	Финансовые рынки
INDUSTRY_CLOSE_END_FUND_DEBT	Закрытый фонд - Долговые инструменты
INDUSTRY_CLOSE_END_FUND_EQUITY	Закрытый фонд - Акции

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
INDUSTRY_CLOSE_END_FUND_FOREIGN	Закрытый фонд - Иностранные
INDUSTRY_CREDIT_SERVICES	Кредитные услуги
INDUSTRY_FINANCIAL_CONGLOMERATE	Финансовые конгломераты
INDUSTRY_FINANCIAL_DATA_EXCHANGE	Финансовые данные и биржи
INDUSTRY_INSURANCE_BROKERS	Страховые брокеры
INDUSTRY_INSURANCE_DIVERSIFIED	Страхование - Диверсифицированные
INDUSTRY_INSURANCE_LIFE	Страхование - Жизнь
INDUSTRY_INSURANCE_PROPERTY	Страхование - Недвижимость и несчастные случаи
INDUSTRY_INSURANCE_REINSURANCE	Страхование - Перестрахование
INDUSTRY_INSURANCE_SPECIALTY	Страхование - Специальное
INDUSTRY_MORTGAGE_FINANCE	Ипотечное финансирование
INDUSTRY_SHELL_COMPANIES	Шелл-компании
Здравоохранение	
INDUSTRY_BIOTECHNOLOGY	Биотехнологии
INDUSTRY_DIAGNOSTICS_RESEARCH	Диагностика и исследования
INDUSTRY_DRUGS_MANUFACTURERS	Фармацевтическое производство - Общее
INDUSTRY_DRUGS_MANUFACTURERS_SPEC	Фармацевтическое производство - Специальное и дженерики
INDUSTRY_HEALTHCARE_PLANS	Планы здравоохранения
INDUSTRY_HEALTH_INFORMATION	Информационные службы здравоохранения
INDUSTRY_MEDICAL_FACILITIES	Медицинские учреждения
INDUSTRY_MEDICAL_DEVICES	Медицинское оборудование
INDUSTRY_MEDICAL_DISTRIBUTION	Медицинские дистрибьюторы
INDUSTRY_MEDICAL_INSTRUMENTS	Медицинские инструменты и расходные материалы
INDUSTRY_PHARM_RETAILERS	Фармацевтические ритейлеры
Промышленность	
INDUSTRY_AEROSPACE_DEFENSE	Аэрокосмическая и оборонная промышленность
INDUSTRY_AIRLINES	Авиакомпании
INDUSTRY_AIRPORTS_SERVICES	Аэропорты и воздушные перевозки

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
INDUSTRY_BUILDING_PRODUCTS	Строительные материалы и оборудование
INDUSTRY_BUSINESS_EQUIPMENT	Деловое оборудование и материалы
INDUSTRY_CONGLOMERATES	Конгломераты
INDUSTRY_CONSULTING_SERVICES	Консалтинговые услуги
INDUSTRY_ELECTRICAL_EQUIPMENT	Электрооборудование и запчасти
INDUSTRY_ENGINEERING_CONSTRUCTION	Инженерное дело и строительство
INDUSTRY_FARM_HEAVY_MACHINERY	Сельскохозяйственное и тяжелое строительное оборудование
INDUSTRY_INDUSTRIAL_DISTRIBUTION	Промышленные дистрибьюторы
INDUSTRY_INFRASTRUCTURE_OPERATIONS	Инфраструктурные операции
INDUSTRY_FREIGHT_LOGISTICS	Интегрированные перевозки и логистика
INDUSTRY_MARINE_SHIPPING	Морские перевозки
INDUSTRY_METAL_FABRICATION	Металлопроизводство
INDUSTRY_POLLUTION_CONTROL	Контроль загрязнения и очистка
INDUSTRY_RAILROADS	Железные дороги
INDUSTRY_RENTAL_LEASING	Аренда и лизинг
INDUSTRY_SECURITY_PROTECTION	Безопасность и защита
INDUSTRY_SPEALITY_BUSINESS_SERVICES	Специализированные бизнес-услуги
INDUSTRY_SPEALITY_MACHINERY	Специализированные промышленные машины
INDUSTRY_STUFFING_EMPLOYMENT	Услуги по трудоустройству
INDUSTRY_TOOLS_ACCESSORIES	Инструменты и инвентарь
INDUSTRY_TRUCKING	Грузоперевозки
INDUSTRY_WASTE_MANAGEMENT	Управление отходами
Недвижимость	
INDUSTRY_REAL_ESTATE_DEVELOPMENT	Недвижимость - Строительство
INDUSTRY_REAL_ESTATE_DIVERSIFIED	Недвижимость - Диверсифицированные
INDUSTRY_REAL_ESTATE_SERVICES	Услуги в сфере недвижимости
INDUSTRY_REIT_DIVERSIFIED	Инвестиционный фонд недвижимости - Диверсифицированные
INDUSTRY_REIT_HEALTCARE	Инвестиционный фонд недвижимости - Медицинские учреждения

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
INDUSTRY_REIT_HOTEL_MOTEL	Инвестиционный фонд недвижимости - Отели
INDUSTRY_REIT_INDUSTRIAL	Инвестиционный фонд недвижимости - Промышленность
INDUSTRY_REIT_MORTAGE	Инвестиционный фонд недвижимости - Ипотека
INDUSTRY_REIT_OFFICE	Инвестиционный фонд недвижимости - Офисы
INDUSTRY_REIT_RESIDENTAL	Инвестиционный фонд недвижимости - Жилые помещения
INDUSTRY_REIT_RETAIL	Инвестиционный фонд недвижимости - Розница
INDUSTRY_REIT_SPECIALITY	Инвестиционный фонд недвижимости - Специальные помещения
Технологии	
INDUSTRY_COMMUNICATION_EQUIPMENT	Коммуникационное оборудование
INDUSTRY_COMPUTER_HARDWARE	Компьютерное оборудование
INDUSTRY_CONSUMER_ELECTRONICS	Бытовая электроника
INDUSTRY_ELECTRONIC_COMPONENTS	Электронные компоненты
INDUSTRY_ELECTRONIC_DISTRIBUTION	Дистрибуция электроники и компьютеров
INDUSTRY_IT_SERVICES	Услуги информационных технологий
INDUSTRY_SCIENTIFIC_INSTRUMENTS	Научно-технические инструменты
INDUSTRY_SEMICONDUCTOR_EQUIPMENT	Полупроводниковое оборудование и материалы
INDUSTRY_SEMICONDUCTORS	Полупроводники
INDUSTRY_SOFTWARE_APPLICATION	Программное обеспечение - Приложения
INDUSTRY_SOFTWARE_INFRASTRUCTURE	Программное обеспечение - Инфраструктура
INDUSTRY_SOLAR	Солнечная энергетика
Коммунальные услуги	
INDUSTRY_UTILITIES_DIVERSIFIED	Коммунальные предприятия - Диверсифицированные
INDUSTRY_UTILITIES_POWERPRODUCERS	Коммунальные предприятия - Независимые производители энергии
INDUSTRY_UTILITIES_RENEWABLE	Коммунальные предприятия - Возобновляемая энергия
INDUSTRY_UTILITIES_REGULATED_ELECTRIC	Коммунальные предприятия - Регулируемые электрические компании

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
INDUSTRY_UTILITIES_REGULATED_GAS	Коммунальные предприятия - Регулируемые газовые компании
INDUSTRY_UTILITIES_REGULATED_WATER	Коммунальные предприятия - Регулируемые водоканалы

© 2000-2025, MetaQuotes Ltd.

Информация о счете

Для получения информации о текущем счете предназначены функции AccountInfoInteger(), AccountInfoDouble() и AccountInfoString(). В качестве параметра эти функции принимают значения из соответствующих перечислений ENUM_ACCOUNT_INFO.

Для функции AccountInfoInteger()

ENUM_ACCOUNT_INFO_INTEGER

Идентификатор	Описание	Тип свойства
ACCOUNT_LOGIN	Номер счета	long
ACCOUNT_TRADE_MODE	Тип торгового счета	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	Размер предоставленного плеча	long
ACCOUNT_LIMIT_ORDERS	Максимально допустимое количество действующих отложенных ордеров	int
ACCOUNT_MARGIN_SO_MODE	Режим задания минимально допустимого уровня залоговых средств	ENUM_ACCOUNT_STOPOUT_MODE
ACCOUNT_TRADE_ALLOWED	Разрешенность торговли для текущего счета	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
ACCOUNT_TRADE_EXPERT	Разрешенность торговли для эксперта	bool
ACCOUNT_MARGIN_MODE	Режим расчета маржи	ENUM_ACCOUNT_MARGIN_MODE
ACCOUNT_CURRENCY_DIGITS	Количество знаков после запятой для валюты счета, необходимых для точного отображения торговых результатов	int
ACCOUNT_FIFO_CLOSE	Признак того, что позиции можно закрывать только по правилу FIFO. Если значение свойства равно true, то позиции по каждому символу разрешается закрывать только в том порядке,	bool

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	в котором они были открыты — сначала самую старую, затем более новую и т.д. При попытке закрыть позиции в ином порядке будет получена ошибка. Для счетов без хеджингового учета позиций (ACCOUNT_MARGIN_MODE!=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) свойство всегда равно false.	
ACCOUNT_HEDGE_ALLOWED	Признак того, что разрешены встречные позиции по одному символу	bool

Для функции AccountInfoDouble()

© 2000-2025, MetaQuotes Ltd.

ENUM_ACCOUNT_INFO_DOUBLE

Идентификатор	Описание	Тип свойства
ACCOUNT_BALANCE	Баланс счета в валюте депозита	double
ACCOUNT_CREDIT	Размер предоставленного кредита в валюте депозита	double
ACCOUNT_PROFIT	Размер текущей прибыли на счете в валюте депозита	double
ACCOUNT_EQUITY	Значение собственных средств на счете в валюте депозита	double
ACCOUNT_MARGIN	Размер зарезервированных залоговых средств на счете в валюте депозита	double
ACCOUNT_MARGIN_FREE	Размер свободных средств на счете в валюте депозита, доступных для открытия позиции	double
ACCOUNT_MARGIN_LEVEL	Уровень залоговых средств	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	на счете в процентах	
ACCOUNT_MARGIN_SO_CALL	Уровень залоговых средств, при котором требуется пополнение счета (Margin Call). В зависимости от установленного ACCOUNT_MARGIN_SO_MODE выражается в процентах либо в валюте депозита	double
ACCOUNT_MARGIN_SO_SO	Уровень залоговых средств, при достижении которого происходит принудительное закрытие самой убыточной позиции (Stop Out). В зависимости от установленного	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
	ACCOUNT_MARGIN_SO_MODE выражается в процентах либо в валюте депозита	
ACCOUNT_MARGIN_INITIAL	Размер средств, зарезервированных на счёте, для обеспечения гарантийной суммы по всем отложенным ордерам	double
ACCOUNT_MARGIN_MAINTENANCE	Размер средств, зарезервированных на счёте, для обеспечения минимальной суммы по всем открытым позициям	double
ACCOUNT_ASSETS	Текущий размер активов на счёте	double
ACCOUNT_LIABILITIES	Текущий размер обязательств на счёте	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип свойства
ACCOUNT_COMMISSION_BLOCKED	Текущая сумма заблокированных комиссий по счёту	double

Для функции AccountInfoString()

ENUM_ACCOUNT_INFO_STRING

Идентификатор	Описание	Тип свойства
ACCOUNT_NAME	Имя клиента	string
ACCOUNT_SERVER	Имя торгового сервера	string
ACCOUNT_CURRENCY	Валюта депозита	string
ACCOUNT_COMPANY	Имя компании, обслуживающей счет	string

Существует несколько видов счетов, которые могут быть открыты на торговом сервере. Для того чтобы узнать тип счета, на котором работает MQL5-программа, предназначено перечисление ENUM_ACCOUNT_TRADE_MODE.

ENUM_ACCOUNT_TRADE_MODE

Идентификатор	Описание
ACCOUNT_TRADE_MODE_DEMO	Демонстрационный торговый счет
ACCOUNT_TRADE_MODE_CONTEST	Конкурсный торговый счет
ACCOUNT_TRADE_MODE_REAL	Реальный торговый счет

При нехватке собственных средств для поддержания открытых позиций возникает ситуация принудительного закрытия Stop Out. Минимальный уровень маржи, при котором наступает Stop Out, может задаваться в процентах или в денежном выражении. Узнать какой режим задан для данного счета можно с помощью перечисления ENUM_ACCOUNT_STOPOUT_MODE.

ENUM_ACCOUNT_STOPOUT_MODE

Идентификатор	Описание
ACCOUNT_STOPOUT_MODE_PERCENT	Уровень задается в процентах

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
ACCOUNT_STOPOUT_MODE_MONEY	Уровень задается в деньгах

ENUM_ACCOUNT_MARGIN_MODE

Идентификатор	Описание
ACCOUNT_MARGIN_MODE_RETAIL_NETTING	Используется для внебиржевого рынка при учете позиций в режиме "неттинг" (по одному символу может быть только одна позиция). Расчет маржи осуществляется на основе типа инструмента (SYMBOL_TRADE_CALC_MODE).
ACCOUNT_MARGIN_MODE_EXCHANGE	Используется для биржевого рынка. Расчет маржи осуществляется на основе дисконтов, указанных в настройках инструментов. Дисконты устанавливаются брокером, однако не могут быть ниже значений, определенных биржей.
ACCOUNT_MARGIN_MODE_RETAIL_HEDGING	Используется для внебиржевого рынка при независимом учете позиций ("хеджинг", по одному символу может быть несколько позиций). Расчет маржи осуществляется на основе типа инструмента (SYMBOL_TRADE_CALC_MODE) и с учетом размера захеджированной маржи (SYMBOL_MARGIN_HEDGED).

Пример скрипта, выводящего краткую информацию о счете.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- имя компании
   string company=AccountInfoString(ACCOUNT_COMPANY);
//--- имя клиента
   string name=AccountInfoString(ACCOUNT_NAME);
//--- номер счета
   long login=AccountInfoInteger(ACCOUNT_LOGIN);
//--- имя сервера
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- валюта счета
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- демо, конкурсный или реальный счет
   ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(AC
//--- теперь превратим значение перечисления в понятный вид
   string trade_mode;
```

© 2000-2025, MetaQuotes Ltd.

```
   switch(account_type)
     {
      case  ACCOUNT_TRADE_MODE_DEMO:
         trade_mode="demo";
         break;
      case  ACCOUNT_TRADE_MODE_CONTEST:
         trade_mode="конкурсный";
         break;
      default:
         trade_mode="реальный";
         break;
     }
//--- в процентах или в денежном выражении задается уровень Stop Out
   ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteg
//--- получим значения уровней, при которых наступает Margin Call и Stop Out
   double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
   double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- выведем краткую информацию по счету
   PrintFormat("Счет клиента '%s' #%d %s открыт в '%s' на сервере '%s'",
               name,login,trade_mode,company,server);
   PrintFormat("Валюта счета - %s, уровень MarginCall и StopOut задается в %s",
               currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"процентах":" в
   PrintFormat("Уровень MarginCall=%G, уровень StopOut=%G",margin_call,stop_out);
  }
```

© 2000-2025, MetaQuotes Ltd.

Статистика тестирования

После окончания тестирования вычисляются статистические показатели результатов торговли по множеству параметров. Значения показателей можно получить с помощью функции TesterStatistics(), указав идентификатор показателя из перечисления ENUM_STATISTICS.

Хотя при вычислении статистики используются показатели двух типов - int и double - функция возвращает все значения в виде double. Все статистические величины, имеющие тип double, по умолчанию выражаются в валюте депозита, если не сказано иное.

ENUM_STATISTICS

Идентификатор	Описание статистического показателя	Тип
STAT_INITIAL_DEPOSIT	Значение начального депозита	double
STAT_WITHDRAWAL	Количество выведенных со счета средств	double
STAT_PROFIT	Чистая прибыль по окончании тестирования, сумма STAT_GROSS_PROFIT и STAT_GROSS_LOSS (STAT_GROSS_LOSS всегда меньше или равно нулю)	double
STAT_GROSS_PROFIT	Общая прибыль, сумма всех прибыльных (положительных) трейдов. Значение больше или равно нулю	double
STAT_GROSS_LOSS	Общий убыток, сумма всех убыточных (отрицательных) трейдов. Значение меньше или равно нулю	double
STAT_MAX_PROFITTRADE	Максимальная прибыль - наибольшее значение среди всех прибыльных трейдов. Значение больше или равно нулю	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание статистического показателя	Тип
STAT_MAX_LOSSTRADE	Максимальный убыток - наименьшее значение среди всех убыточных трейдов. Значение меньше или равно нулю	double
STAT_CONPROFITMAX	Максимальная прибыль в последовательности прибыльных трейдов. Значение больше или равно нулю	double
STAT_CONPROFITMAX_TRADES	Количество трейдов, сформировавших STAT_CONPROFITMAX (максимальная прибыль в последовательности прибыльных трейдов)	int
STAT_MAX_CONWINS	Общая прибыль в самой длинной серии прибыльных трейдов	double
STAT_MAX_CONPROFIT_TRADES	Количество трейдов в самой длинной серии прибыльных трейдов STAT_MAX_CONWINS	int
STAT_CONLOSSMAX	Максимальный убыток в последовательности убыточных трейдов. Значение меньше или равно нулю	double
STAT_CONLOSSMAX_TRADES	Количество трейдов, сформировавших STAT_CONLOSSMAX (максимальный убыток в последовательности убыточных трейдов)	int
STAT_MAX_CONLOSSES	Общий убыток в самой длинной серии убыточных трейдов	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание статистического показателя	Тип
STAT_MAX_CONLOSS_TRADES	Количество трейдов в самой длинной серии убыточных трейдов STAT_MAX_CONLOSSES	int
STAT_BALANCEMIN	Минимальное значение баланса	double
STAT_BALANCE_DD	Максимальная просадка баланса в деньгах. В процессе торговли баланс может испытать множество просадок, берется наибольшее значение.	double
STAT_BALANCEDD_PERCENT	Просадка баланса в процентах, которая была зафиксирована в момент максимальной просадки баланса в деньгах (STAT_BALANCE_DD).	double
STAT_BALANCE_DDREL_PERCENT	Максимальная просадка баланса в процентах. В процессе торговли баланс может испытать множество просадок, для каждой фиксируется относительное значение просадки в процентах. Возвращается наибольшее значение	double
STAT_BALANCE_DD_RELATIVE	Просадка баланса в деньгах, которая была зафиксирована в момент максимальной просадки баланса в процентах	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание статистического показателя	Тип
	(STAT_BALANCE_DDREL_PERCENT).	
STAT_EQUITYMIN	Минимальное значение собственных средств	double
STAT_EQUITY_DD	Максимальная просадка средств в деньгах. В процессе торговли средства могут испытать множество просадок, берется наибольшее значение.	double
STAT_EQUITYDD_PERCENT	Просадка средств в процентах, которая была зафиксирована в момент максимальной просадки средств в деньгах (STAT_EQUITY_DD).	double
STAT_EQUITY_DDREL_PERCENT	Максимальная просадка средств в процентах. В процессе торговли средства могут испытать множество просадок, для каждой фиксируется относительное значение просадки в процентах. Возвращается наибольшее значение	double
STAT_EQUITY_DD_RELATIVE	Просадка средств в деньгах, которая была зафиксирована в момент максимальной просадки средств в процентах (STAT_EQUITY_DDREL_PERCENT).	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание статистического показателя	Тип
STAT_EXPECTED_PAYOFF	Математическое ожидание выигрыша	double
STAT_PROFIT_FACTOR	Прибыльность - отношение STAT_GROSS_PROFIT /STAT_GROSS_LOSS. Если STAT_GROSS_LOSS=0 , то прибыльность принимает значение DBL_MAX	double
STAT_RECOVERY_FACTOR	Фактор восстановления - отношение STAT_PROFIT/STAT_ BALANCE_DD	double
STAT_SHARPE_RATIO	Коэффициент Шарпа	double
STAT_MIN_MARGINLEVEL	Минимальное достигнутое значение уровня маржи	double
STAT_CUSTOM_ONTESTER	Значение рассчитанного пользовательского критерия оптимизации, возвращаемого функцией OnTester()	double
STAT_DEALS	Количество совершенных сделок	int
STAT_TRADES	Количество трейдов	int
STAT_PROFIT_TRADES	Прибыльные трейды	int
STAT_LOSS_TRADES	Убыточные трейды	int
STAT_SHORT_TRADES	Короткие трейды	int
STAT_LONG_TRADES	Длинные трейды	int
STAT_PROFIT_SHORTTRADES	Короткие прибыльные трейды	int
STAT_PROFIT_LONGTRADES	Длинные прибыльные трейды	int

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание статистического показателя	Тип
STAT_PROFITTRADES_AVGCON	Средняя длина прибыльной серии трейдов	int
STAT_LOSSTRADES_AVGCON	Средняя длина убыточной серии трейдов	int
STAT_COMPLEX_CRITERION	Комплексный критерий оптимизации	double

© 2000-2025, MetaQuotes Ltd.

Торговые константы

Разнообразные константы, используемые для программирования торговых стратегий, разделены на следующие группы:

- Информация об исторических данных по инструменту – получение общей информации по финансовому инструменту;
- Свойства ордеров – получение информации о торговых ордерах;
- Свойства позиций – получение информации о текущих позициях;
- Свойства сделок – получение информации о совершенных сделках;
- Типы торговых операций – описание доступных торговых операций;
- Типы торговых транзакций - описание возможных типов торговых транзакций;
- Виды заявок в стакане цен - для разделения заявок по направлению запрашиваемой торговой операции.

© 2000-2025, MetaQuotes Ltd.

Информация об исторических данных по инструменту

При доступе к таймсериям для получения дополнительной информации об инструменте используется функция SeriesInfoInteger(). В качестве параметра этой функции передается идентификатор требуемого свойства, который может принимать одно из значений перечисления ENUM_SERIES_INFO_INTEGER.

ENUM_SERIES_INFO_INTEGER

Идентификатор	Описание	Тип свойства
SERIES_BARS_COUNT	Количество баров по символу-периоду на данный момент	long
SERIES_FIRSTDATE	Самая первая дата по символу-периоду на данный момент	datetime
SERIES_LASTBAR_DATE	Время открытия последнего бара по символу-периоду	datetime
SERIES_SERVER_FIRSTDATE	Самая первая дата в истории по символу на сервере независимо от периода	datetime
SERIES_TERMINAL_FIRSTDATE	Самая первая дата в истории по символу в клиентском терминале независимо от периода	datetime
SERIES_SYNCHRONIZED	Признак синхронизированности данных по символу/периоду на данный момент	bool

© 2000-2025, MetaQuotes Ltd.

Свойства ордеров

Приказы на проведение торговых операций оформляются ордерами. Каждый ордер имеет множество свойств для чтения, информацию по ним можно получать с помощью функций OrderGet...() и HistoryOrderGet...().

Для функций OrderGetInteger() и HistoryOrderGetInteger()

ENUM_ORDER_PROPERTY_INTEGER

Идентификатор	Описание	Тип
ORDER_TICKET	Тикет ордера. Уникальное число, которое присваивается каждому ордеру	long
ORDER_TIME_SETUP	Время постановки ордера	datetime
ORDER_TYPE	Тип ордера	ENUM_ORDER_TYPE
ORDER_STATE	Статус ордера	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	Время истечения ордера	datetime
ORDER_TIME_DONE	Время исполнения или снятия ордера	datetime
ORDER_TIME_SETUP_MSC	Время установки ордера на исполнение в миллисекундах с 01.01.1970	long
ORDER_TIME_DONE_MSC	Время исполнения/снятия ордера в миллисекундах с 01.01.1970	long

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип
ORDER_TYPE_FILLING	Тип исполнения по остатку	ENUM_ORDER_TYPE_FILLING
ORDER_TYPE_TIME	Время жизни ордера	ENUM_ORDER_TYPE_TIME
ORDER_MAGIC	Идентификатор эксперта выставившего ордер (предназначен для того, чтобы каждый эксперт выставлял свой собственный уникальный номер)	long
ORDER_REASON	Причина или источник выставления ордера	ENUM_ORDER_REASON
ORDER_POSITION_ID	Идентификатор позиции, который ставится на ордере при его исполнении. Каждый исполненный ордер порождает сделку, которая открывает новую или изменяет уже существующую позицию. Идентификатор этой позиции и устанавливается	long

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип
	исполненному ордеру в этот момент.	
ORDER_POSITION_BY_ID	Идентификатор встречной позиции для ордеров типа ORDER_TYPE_CLOSE_BY.	long

Для функций OrderGetDouble() и HistoryOrderGetDouble()

ENUM_ORDER_PROPERTY_DOUBLE

Идентификатор	Описание	Тип
ORDER_VOLUME_INITIAL	Первоначальный объем при постановке ордера	double
ORDER_VOLUME_CURRENT	Невыполненный объем	double
ORDER_PRICE_OPEN	Цена, указанная в ордере	double
ORDER_SL	Уровень Stop Loss	double
ORDER_TP	Уровень Take Profit	double
ORDER_PRICE_CURRENT	Текущая цена по символу ордера	double
ORDER_PRICE_STOPLIMIT	Цена постановки Limit ордера при срабатывании StopLimit ордера	double

Для функций OrderGetString() и HistoryOrderGetString()

ENUM_ORDER_PROPERTY_STRING

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип
ORDER_SYMBOL	Символ, по которому выставлен ордер	string
ORDER_COMMENT	Комментарий	string
ORDER_EXTERNAL_ID	Идентификатор ордера во внешней торговой системе (на бирже)	string

При отправке торгового запроса функцией OrderSend() для некоторых операций необходимо указать тип ордера. Тип ордера указывается в поле *type* специальной структуры MqlTradeRequest, и может принимать значения из перечисления ENUM_ORDER_TYPE.

ENUM_ORDER_TYPE

Идентификатор	Описание
ORDER_TYPE_BUY	Рыночный ордер на покупку
ORDER_TYPE_SELL	Рыночный ордер на продажу
ORDER_TYPE_BUY_LIMIT	Отложенный ордер Buy Limit
ORDER_TYPE_SELL_LIMIT	Отложенный ордер Sell Limit
ORDER_TYPE_BUY_STOP	Отложенный ордер Buy Stop
ORDER_TYPE_SELL_STOP	Отложенный ордер Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	По достижении цены ордера выставляется отложенный ордер Buy Limit по цене StopLimit
ORDER_TYPE_SELL_STOP_LIMIT	По достижении цены ордера выставляется отложенный ордер Sell Limit по цене StopLimit
ORDER_TYPE_CLOSE_BY	Ордер на закрытие позиции встречной позицией

Каждый ордер имеет статус, описывающий его состояние. Для получения информации используйте функцию OrderGetInteger() или HistoryOrderGetInteger() с модификатором ORDER_STATE. Допустимые значения хранятся в перечислении ENUM_ORDER_STATE.

ENUM_ORDER_STATE

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
ORDER_STATE_STARTED	Ордер проверен на корректность, но еще не принят брокером
ORDER_STATE_PLACED	Ордер принят
ORDER_STATE_CANCELED	Ордер снят клиентом
ORDER_STATE_PARTIAL	Ордер выполнен частично
ORDER_STATE_FILLED	Ордер выполнен полностью
ORDER_STATE_REJECTED	Ордер отклонен
ORDER_STATE_EXPIRED	Ордер снят по истечении срока его действия
ORDER_STATE_REQUEST_ADD	Ордер в состоянии регистрации (выставление в торговую систему)
ORDER_STATE_REQUEST_MODIFY	Ордер в состоянии модификации (изменение его параметров)
ORDER_STATE_REQUEST_CANCEL	Ордер в состоянии удаления (удаление из торговой системы)

При отправке торгового запроса на исполнение **в действующий момент времени** (time in force) нужно указать цену и требуемый объем покупки/продажи. При этом необходимо учитывать, что на финансовых рынках нет гарантии того, что в данный момент времени по данному финансовому инструменту доступен весь запрашиваемый объем по желаемой цене. Поэтому проведение торговых операций в реальном времени регулируется с помощью режимов исполнения по цене и объему. Режимы, или по-другому политики исполнения, определяют правила для случаев, когда изменилась цена или запрашиваемый объем нельзя выполнить полностью **в текущий момент времени**.

Режим исполнения по цене - его можно получить из свойства символа SYMBOL_TRADE_EXEMODE, которое содержит комбинацию флагов из перечисления ENUM_SYMBOL_TRADE_EXECUTION.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
Режим исполнения (Request Execution)	Исполнение рыночного ордера по цене, предварительно полученной	SYMBOL_TRADE_EXECUTION_REQUEST

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	от брокера. Перед отправкой рыночного ордера у брокера запрашиваются цены его исполнения. После их получения выполнение ордера по данной цене можно либо подтвердить, либо отклонить.	
Немедленное исполнение (Instant Execution)	Немедленное исполнение рыночного ордера по указанной цене.	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	При отправке торгового запроса на исполнение платформа автоматически подставляет в ордер текущие цены. • Если брокер принимает цену, то о	

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	рдербудетисполнен. • Еслиброкернеприниматзапрошенн	

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	ую цену, то про исходит так называемое "реквотирование"	

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	(Request) – брокер возвращает цены, по которым может быть

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	ьиспользованный ордер.	
Исполнение по рынку (Market Execution)	Решение о цене исполнения принимает брокер без дополнительного согласования с трейдером. Отправка рыночного ордера в таком режим	SYMBOL_TRADE_EXECUTION_MARKET

© 2000-2025, MetaQuotes Ltd.

Режим исполнения	Описание	Значение в ENUM_SYMBOL_TRADE_EXECUTION
	е подразумевает досрочное согласие с ценой, по которой он будет выполнен.	
Биржевое исполнение (Exchange Execution)	Торговые операции совершаются по ценам текущих рыночных предложений.	SYMBOL_TRADE_EXECUTION_EXCHANGE

Политика заполнения объема - указывается в свойстве ордера ORDER_TYPE_FILLING и может содержать только значения из перечисления ENUM_ORDER_TYPE_FILLING

Политика исполнения	Описание	Значение в ENUM_ORDER_TYPE_FILLING
Все/Ничего (Fill or Kill)	Ордер может быть исполнен исключительно в указанном объеме.	ORDER_FILLING_FOK

© 2000-2025, MetaQuotes Ltd.

Политика исполнения	Описание	Значение в ENUM_ORDER_TYPE_FILLING
	Если на рынке в данный момент не присутствует достаточного объема финансового инструмента, то ордер не будет исполнен. Необходимый объем может быть составлен из нескольких предложений, доступных в данный момент на рынке.	

© 2000-2025, MetaQuotes Ltd.

Политика исполнения	Описание	Значение в ENUM_ORDER_TYPE_FILLING
	Возможность использования FOK-ордеров определяется на торговом сервере.	
Все/Частично (Immediate or Cancel)	Трейдер соглашается совершить сделку по максимально доступному на рынке объему в пределах указанного в ордере. В случае невозможности полного исполн	ORDER_FILLING_IOC

© 2000-2025, MetaQuotes Ltd.

Политика исполнения	Описание	Значение в ENUM_ORDER_TYPE_FILLING
	ения ордер будет исполнен на доступный объем, а неисполненный объем ордера будет отменен. Возможность использования IOC-ордеров определяется на торговом сервере.	
Пассивная (Book or Cancel)	BoC-ордер предполагает, что ордер может быть только выставлен в стакан цен и	ORDER_FILLING_BOC

© 2000-2025, MetaQuotes Ltd.

Политика исполнения	Описание	Значение в ENUM_ORDER_TYPE_FILLING
	не может быть немедленно исполнен. Если ордер может быть исполнен немедленно при выставлении, то он снимается. Фактически, политика BOC гарантирует, что цена выставленного ордера будет хуже текущего рынка. BoC-ордеры используются для реализации	

© 2000-2025, MetaQuotes Ltd.

Политика исполнения	Описание	Значение в ENUM_ORDER_TYPE_FILLING
	пассивной торговли, чтобы ордер гарантированно не был исполнен сразу при выставлении и не повлиял на текущую ликвидность. Поддерживается только для лимитных и стоп-лимитных ордеров (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE	

© 2000-2025, MetaQuotes Ltd.

Политика исполнения	Описание	Значение в ENUM_ORDER_TYPE_FILLING
	_BUY_STOP_LIMIT, ORDER_TYPE_SELL_STOP_LIMIT).	
Вернуть (Return)	В случае частичного исполнения ордер с остаточным объемом не снимается, а продолжает действовать. Return-ордеры не разрешены в режиме исполнения Market Execution (исполнение ордеров по рынку —	ORDER_FILLING_RETURN

© 2000-2025, MetaQuotes Ltd.

Политика исполнения	Описание	Значение в ENUM_ORDER_TYPE_FILLING
	SYMBOL_TRADE_EXECUTION_MARKET).	

При отправке торгового запроса функцией OrderSend() нужную политику исполнения по объему можно задать в поле *type_filling* в специальной структуре MqlTradeRequest, допустимы значения из перечисления ENUM_ORDER_TYPE_FILLING. Для получения значения этого свойства в конкретном действующем/отработавшем ордере используйте функцию OrderGetInteger() или HistoryOrderGetInteger() с модификатором ORDER_TYPE_FILLING.

Перед отправкой ордера с исполнением в текущий момент времени, для правильного выставления значения ORDER_TYPE_FILLING (тип исполнения по объему), для каждого финансового инструмента с помощью функции SymbolInfoInteger() можно получить значение свойства SYMBOL_FILLING_MODE, которое показывает разрешенные для этого символа типы заполнения по объему в виде комбинации флагов. При этом нужно отметить, что тип заливки ORDER_FILLING_RETURN разрешен всегда, кроме режима "Исполнение по рынку" (SYMBOL_TRADE_EXECUTION_MARKET).

Применение типов заливки в зависимости от режима исполнения можно представить в виде таблицы:

Режим исполнения\Политика заполнения	Все/Ничего (FOK ORDER_FILLING_FOK)	Все/Частично (IOC ORDER_FILLING_IOC)	Вернуть (Return ORDER_FILLING_RETURN)
Немедленное исполнение (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (независимо от настройки символа)	+ (независимо от настройки символа)	+ (всегда)
Исполнения по запросу SYMBOL_TRADE_EXECUTION_REQUEST	+ (независимо от настройки символа)	+ (независимо от настройки символа)	+ (всегда)
Исполнение по рынку SYMBOL_TRADE_EXECUTION_MARKET	+ (задается в настройках символа)	+ (задается в настройках символа)	- (запрещено при любых настройках символа)
Биржевое исполнение SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (задается в настройках символа)	+ (задается в настройках символа)	+ (всегда)

Для отложенных ордеров независимо от режима исполнения (SYMBOL_TRADE_EXEMODE) необходимо использовать тип заливки ORDER_FILLING_RETURN, так как такие ордеры не предназначены для исполнения в момент отправки. При использовании отложенных ордеров

© 2000-2025, MetaQuotes Ltd.

трейдер заранее соглашается, что при наступлении условий для проведения сделки по данному ордеру, брокером будет использован тот тип заливки, который поддерживается данной торговой площадкой.

Срок действия ордера можно задать в поле *type_time* специальной структуры MqlTradeRequest при отправке торгового запроса функцией OrderSend(). Допустимы значения из перечисления ENUM_ORDER_TYPE_TIME. Для получения значения этого свойства используйте функцию OrderGetInteger() или HistoryOrderGetInteger() с модификатором ORDER_TYPE_TIME.

ENUM_ORDER_TYPE_TIME

Идентификатор	Описание
ORDER_TIME_GTC	Ордер будет находится в очереди до тех пор, пока не будет снят
ORDER_TIME_DAY	Ордер будет действовать только в течение текущего торгового дня
ORDER_TIME_SPECIFIED	Ордер будет действовать до даты истечения
ORDER_TIME_SPECIFIED_DAY	Ордер будет действовать до 23:59:59 указанного дня. Если это время не попадает на торговую сессию, истечение наступит в ближайшее торговое время.

В свойстве ORDER_REASON содержится причина выставления ордера. Ордер может быть выставлен с помощью MQL5 программы, или из мобильного приложения, или в результате наступления события StopOut, и т.д. Возможные значения ORDER_REASON описываются в перечислении ENUM_ORDER_REASON.

ENUM_ORDER_REASON

Идентификатор	Описание
ORDER_REASON_CLIENT	Ордер выставлен из десктопного терминала
ORDER_REASON_MOBILE	Ордер выставлен из мобильного приложения
ORDER_REASON_WEB	Ордер выставлен из веб-платформы
ORDER_REASON_EXPERT	Ордер выставлен из MQL5-программы – советником или скриптом
ORDER_REASON_SL	Ордер выставлен в результате срабатывания Stop Loss
ORDER_REASON_TP	Ордер выставлен в результате срабатывания Take Profit
ORDER_REASON_SO	Ордер выставлен в результате наступления события Stop Out

© 2000-2025, MetaQuotes Ltd.

Свойства позиций

Результатом совершения торговых операций являются открытие позиции, изменение её объема и/или направления, или ее ликвидация. Торговые операции проводятся на основание ордеров, отправляемых функцией OrderSend() в виде торговых запросов. Для каждого финансового инструмента (символа) возможна только одна открытая позиция. Позиция имеет набор свойств, доступных для чтений функциями PositionGet...().

Для функции PositionGetInteger()

ENUM_POSITION_PROPERTY_INTEGER

Идентификатор	Описание	Тип
POSITION_TICKET	Тикет позиции. Уникальное число, которое присваивается каждой вновь открытой позиции. Как правило, соответствует тикету ордера, в результате которого она была открыта, за исключением случаев изменения тикета в результате служебных операций на сервере. Например, начисления свопов переоткрытием позиции. Для нахождения ордера, которым была открыта позиция, следует использовать свойство POSITION_IDENTIFIER.	long

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип
	Значение POSITION_TICKET соответствует MqlTradeRequest::position.	
POSITION_TIME	Время открытия позиции	datetime
POSITION_TIME_MSC	Время открытия позиции в миллисекундах с 01.01.1970	long
POSITION_TIME_UPDATE	Время изменения позиции	datetime
POSITION_TIME_UPDATE_MSC	Время изменения позиции в миллисекундах с 01.01.1970	long
POSITION_TYPE	Тип позиции	ENUM_POSITION_TYPE
POSITION_MAGIC	Magic number для позиции (смотри ORDER_MAGIC)	long
POSITION_IDENTIFIER	Идентификатор позиции - это уникальное число, которое присваивается каждой вновь открытой позиции и не изменяется в течение всей ее жизни. Соответствует тикету ордера, которым была открыта позиция. Идентификатор позиции указывается в каждом ордере	long

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип
	(ORDER_POSITION_ID) и сделке (DEAL_POSITION_ID), которая ее открыла, изменила или закрыла. Используйте это свойство для поиска ордеров и сделок, связанных с позицией. При развороте позиции в режиме неттинга (единой сделкой in/out) идентификатор позиции POSITION_IDENTIFIER не изменяется. Однако при этом POSITION_TICKET изменяется на тикет ордера, в результате которого произошел разворот. В режиме хеджинга разворот позиции не предусмотрен.	
POSITION_REASON	Причина открытия позиции	ENUM_POSITION_REASON

Для функции PositionGetDouble()

ENUM_POSITION_PROPERTY_DOUBLE

Идентификатор	Описание	Тип
POSITION_VOLUME	Объем позиции	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип
POSITION_PRICE_OPEN	Цена позиции	double
POSITION_SL	Уровень Stop Loss для открытой позиции	double
POSITION_TP	Уровень Take Profit для открытой позиции	double
POSITION_PRICE_CURRENT	Текущая цена по символу	double
POSITION_SWAP	Накопленный своп	double
POSITION_PROFIT	Текущая прибыль	double

Для функции PositionGetString()

ENUM_POSITION_PROPERTY_STRING

Идентификатор	Описание	Тип
POSITION_SYMBOL	Символ, по которому открыта позиция	string
POSITION_COMMENT	Комментарий к позиции	string
POSITION_EXTERNAL_ID	Идентификатор позиции во внешней системе (на бирже)	string

Направление открытой позиции (покупка или продажа) определяется значением из перечисления ENUM_POSITION_TYPE. Для получения типа открытой позиции используйте функцию PositionGetInteger() с модификатором POSITION_TYPE.

ENUM_POSITION_TYPE

Идентификатор	Описание
POSITION_TYPE_BUY	Покупка
POSITION_TYPE_SELL	Продажа

© 2000-2025, MetaQuotes Ltd.

В свойстве POSITION_REASON содержится причина открытия позиции. Позиция может быть открыта в результате срабатывания ордера, который был выставлен из десктопного терминала, из мобильного приложения, с помощью советника и т.д. Возможные значения POSITION_REASON описываются в перечислении ENUM_POSITION_REASON.

ENUM_POSITION_REASON

Идентификатор	Описание
POSITION_REASON_CLIENT	Позиция открыта в результате срабатывания ордера, выставленного из десктопного терминала
POSITION_REASON_MOBILE	Позиция открыта в результате срабатывания ордера, выставленного из мобильного приложения
POSITION_REASON_WEB	Позиция открыта в результате срабатывания ордера, выставленного из веб-платформы
POSITION_REASON_EXPERT	Позиция открыта в результате срабатывания ордера, выставленного из MQL5-программы – советником или скриптом

© 2000-2025, MetaQuotes Ltd.

Свойства сделок

Сделка является отражением факта совершения торговой операции на основании ордера, содержащего торговый приказ. Каждая сделка описывается свойствами, позволяющими получить информацию о ней. Для чтения значений свойств используются функции вида HistoryDealGet...(), возвращающие значения из соответствующих перечислений.

Для функции HistoryDealGetInteger()

ENUM_DEAL_PROPERTY_INTEGER

Идентификатор	Описание	Тип
DEAL_TICKET	Тикет сделки. Уникальное число, которое присваивается каждой сделке	long
DEAL_ORDER	Ордер, на основание которого выполнена сделка	long
DEAL_TIME	Время совершения сделки	datetime
DEAL_TIME_MSC	Время совершения сделки в миллисекундах с 01.01.1970	long
DEAL_TYPE	Тип сделки	ENUM_DEAL_TYPE
DEAL_ENTRY	Направление сделки - вход в рынок, выход из рынка или разворот	ENUM_DEAL_ENTRY
DEAL_MAGIC	Magic number для сделки (смотри ORDER_MAGIC)	long
DEAL_REASON	Причина или источник проведения сделки	ENUM_DEAL_REASON
DEAL_POSITION_ID	Идентификатор позиции, в открытии, изменении или закрытии которой участвовала эта сделка. Каждая позиция имеет уникальный идентификатор, который присваивается всем сделкам, совершенным на инструменте в течение всей жизни позиции.	long

Для функции HistoryDealGetDouble()

ENUM_DEAL_PROPERTY_DOUBLE

Идентификатор	Описание	Тип
DEAL_VOLUME	Объем сделки	double
DEAL_PRICE	Цена сделки	double
DEAL_COMMISSION	Комиссия по сделке	double
DEAL_SWAP	Накопленный своп при закрытии	double

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип
DEAL_PROFIT	Финансовый результат сделки	double
DEAL_FEE	Оплата за проведение сделки, начисляется сразу после совершения сделки	double
DEAL_SL	Уровень Stop Loss • Для сделки входа или разворота берется значение Stop Loss из ордера, которым была открыта или развернута позиция • Для сделки выхода берется значение Stop Loss из позиции на момент её закрытия	double
DEAL_TP	Уровень Take Profit • Для сделки входа или разворота берется значение Take Profit из ордера, которым была открыта или развернута позиция • Для сделки выхода берется значение Take Profit из позиции на момент её закрытия	double

Для функции HistoryDealGetString()

ENUM_DEAL_PROPERTY_STRING

Идентификатор	Описание	Тип
DEAL_SYMBOL	Имя символа, по которому произведена сделка	string
DEAL_COMMENT	Комментарий к сделке	string
DEAL_EXTERNAL_ID	Идентификатор сделки во внешней торговой системе (на бирже)	string

Каждая сделка характеризуется типом, возможные значения перечислены в ENUM_DEAL_TYPE. Для получения информации о типе сделки используйте функцию HistoryDealGetInteger() с модификатором DEAL_TYPE.

ENUM_DEAL_TYPE

Идентификатор	Описание
DEAL_TYPE_BUY	Покупка
DEAL_TYPE_SELL	Продажа
DEAL_TYPE_BALANCE	Начисление баланса

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
DEAL_TYPE_CREDIT	Начисление кредита
DEAL_TYPE_CHARGE	Дополнительные сборы
DEAL_TYPE_CORRECTION	Корректирующая запись
DEAL_TYPE_BONUS	Перечисление бонусов
DEAL_TYPE_COMMISSION	Дополнительные комиссии
DEAL_TYPE_COMMISSION_DAILY	Комиссия, начисляемая в конце торгового дня
DEAL_TYPE_COMMISSION_MONTHLY	Комиссия, начисляемая в конце месяца
DEAL_TYPE_COMMISSION_AGENT_DAILY	Агентская комиссия, начисляемая в конце торгового дня
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Агентская комиссия, начисляемая в конце месяца
DEAL_TYPE_INTEREST	Начисления процентов на свободные средства
DEAL_TYPE_BUY_CANCELED	Отмененная сделка покупки. Возможная ситуация, когда ранее совершенная сделка на покупку отменяется. В таком случае тип ранее совершенной сделки (DEAL_TYPE_BUY) меняется на DEAL_TYPE_BUY_CANCELED, а ее прибыль/убыток обнуляется. Ранее полученная прибыль/убыток начисляется/списывается со счета отдельной балансовой операцией
DEAL_TYPE_SELL_CANCELED	Отмененная сделка продажи. Возможная ситуация, когда ранее совершенная сделка на продажу отменяется. В таком случае тип ранее совершенной сделки (DEAL_TYPE_SELL) меняется на DEAL_TYPE_SELL_CANCELED, а ее прибыль/убыток обнуляется. Ранее полученная прибыль/убыток начисляется/списывается со счета отдельной балансовой операцией
DEAL_DIVIDEND	Начисление дивиденда
DEAL_DIVIDEND_FRANKED	Начисление франкированного дивиденда (освобожденного от уплаты налога)
DEAL_TAX	Начисление налога

Сделки различаются не только по типу, задаваемого в перечислении ENUM_DEAL_TYPE, но и по способу изменения позиции. Это может быть простое открытие позиции или наращивание объема ранее открытой позиции (вход в рынок), закрытие позиции сделкой противоположного направления соответствующим объемом (выход их рынка) или переворот позиции в том случае, когда объем сделки в противоположном направлении перекрывает объем ранее открытой позиции.

© 2000-2025, MetaQuotes Ltd.

Все эти ситуации описаны значениями из перечисления ENUM_DEAL_ENTRY. Для получения этой информации о сделке используйте функцию HistoryDealGetInteger() с модификатором DEAL_ENTRY.

ENUM_DEAL_ENTRY

Идентификатор	Описание
DEAL_ENTRY_IN	Вход в рынок
DEAL_ENTRY_OUT	Выход из рынка
DEAL_ENTRY_INOUT	Разворот
DEAL_ENTRY_OUT_BY	Закрытие встречной позицией

В свойстве DEAL_REASON содержится причина проведения сделки. Сделка может быть проведена в результате срабатывания ордера, выставленного из мобильного приложения или из MQL5 программы; либо в результате наступления события StopOut или начисления/списания вариационной маржи, и т.д. Возможные значения DEAL_REASON описываются в перечислении ENUM_DEAL_REASON. Для неторговых сделок, вызванных операциями изменения баланса, кредита, начисления комиссий и прочих, в качестве причины указывается DEAL_REASON_CLIENT.

ENUM_DEAL_REASON

Идентификатор	Описание
DEAL_REASON_CLIENT	Сделка проведена в результате срабатывания ордера, выставленного из десктопного терминала
DEAL_REASON_MOBILE	Сделка проведена в результате срабатывания ордера, выставленного из мобильного приложения
DEAL_REASON_WEB	Сделка проведена в результате срабатывания ордера, выставленного из веб-платформы
DEAL_REASON_EXPERT	Сделка проведена в результате срабатывания ордера, выставленного из MQL5-программы – советником или скриптом
DEAL_REASON_SL	Сделка проведена в результате срабатывания ордера Stop Loss
DEAL_REASON_TP	Сделка проведена в результате срабатывания ордера Take Profit
DEAL_REASON_SO	Сделка проведена в результате наступления события Stop Out
DEAL_REASON_ROLLOVER	Сделка проведена по причине переноса позиции
DEAL_REASON_VMARGIN	Сделка проведена по причине начисления/списания вариационной маржи
DEAL_REASON_SPLIT	Сделка проведена по причине сплита (понижения цены) инструмента, по которому имелась позиция на момент проведения сплита
DEAL_REASON_CORPORATE_ACTION	Сделка проведена в результате корпоративного действия: объединения или переименования бумаги, переноса клиента ну

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
	другой счет и т.д.

© 2000-2025, MetaQuotes Ltd.

Типы торговых операций

Торговля осуществляется посредством отправки с помощью функции OrderSend() приказов на открытие позиций, а также приказов на установку, модификацию и удаление отложенных ордеров. Каждый торговый приказ содержит указание на тип запрашиваемой торговой операции. Торговые операции описаны в перечислении ENUM_TRADE_REQUEST_ACTIONS.

ENUM_TRADE_REQUEST_ACTIONS

Идентификатор	Описание
TRADE_ACTION_DEAL	Установить торговый ордер на немедленное совершение сделки с указанными параметрами (поставить рыночный ордер)
TRADE_ACTION_PENDING	Установить торговый ордер на совершение сделки при указанных условиях (отложенный ордер)
TRADE_ACTION_SLTP	Изменить значения Stop Loss и Take Profit у открытой позиции
TRADE_ACTION_MODIFY	Изменить параметры ранее установленного торгового ордера
TRADE_ACTION_REMOVE	Удалить ранее выставленный отложенный торговый ордер
TRADE_ACTION_CLOSE_BY	Закрыть позицию встречной

Пример торговой операции TRADE_ACTION_DEAL для открытия позиции Buy:

```
#define EXPERT_MAGIC 123456   // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Открытие позиции Buy                                             |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- параметры запроса
   request.action   =TRADE_ACTION_DEAL;                     // тип торговой операции
   request.symbol   =Symbol();                              // символ
   request.volume   =0.1;                                   // объем в 0.1 лот
   request.type     =ORDER_TYPE_BUY;                        // тип ордера
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // цена для открытия
   request.deviation=5;                                     // допустимое отклонение
   request.magic    =EXPERT_MAGIC;                          // MagicNumber ордера
//--- отправка запроса
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // если отправить запрос
//--- информация об операции
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Пример торговой операции TRADE_ACTION_DEAL для открытия позиции Sell:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Открытие позиции Sell                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- параметры запроса
   request.action   =TRADE_ACTION_DEAL;                     // тип торговой операции
   request.symbol   =Symbol();                              // символ
   request.volume   =0.2;                                   // объем в 0.2 лот
   request.type     =ORDER_TYPE_SELL;                       // тип ордера
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // цена для открытия
   request.deviation=5;                                     // допустимое отклонение
   request.magic    =EXPERT_MAGIC;                          // MagicNumber ордера
//--- отправка запроса
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // если отправить запрос
//--- информация об операции
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Пример торговой операции TRADE_ACTION_DEAL для закрытия позиций:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Закрытие всех позиций                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление запроса и результата
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // количество открытых позиций
//--- перебор всех открытых позиций
   for(int i=total-1; i>=0; i--)
     {
      //--- параметры ордера
      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);
      //--- вывод информации о позиции
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- если MagicNumber совпадает
      if(magic==EXPERT_MAGIC)
        {
         //--- обнуление значений запроса и результата
         ZeroMemory(request);
         ZeroMemory(result);
         //--- установка параметров операции
         request.action   =TRADE_ACTION_DEAL;        // тип торговой операции
         request.position =position_ticket;          // тикет позиции
         request.symbol   =position_symbol;          // символ
         request.volume   =volume;                   // объем позиции
         request.deviation=5;                        // допустимое отклонение от цены
         request.magic    =EXPERT_MAGIC;             // MagicNumber позиции
         //--- установка цены и типа ордера в зависимости от типа позиции
         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;
           }
         //--- вывод информации о закрытии
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToString
         //--- отправка запроса
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // если отправить запро
         //--- информация об операции
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
```

Пример торговой операции TRADE_ACTION_PENDING для установки отложенного ордера:

```
#property description "Пример установки отложенных ордеров"
#property script_show_inputs
#define EXPERT_MAGIC 123456                                  // MagicNumber эксперта
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;    // тип ордера
//+------------------------------------------------------------------+
//| Установка отложенных ордеров                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- параметры для установки отложенного ордера
   request.action   =TRADE_ACTION_PENDING;                                  // тип торговой
   request.symbol   =Symbol();                                              // символ
   request.volume   =0.1;                                                   // объем в 0.1
   request.deviation=2;                                                     // допустимое
   request.magic    =EXPERT_MAGIC;                                          // MagicNumber
   int offset = 50;                                                         // отступ от те
   double price;                                                            // цена срабать
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);                     // размер пункт
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);                     // кол-во знако
   //--- проверка типа операции
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                               // тип ордера
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;             // цена для отк
      request.price    =NormalizeDouble(price,digits);                      // нормализован
     }
   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);                      // нормализова
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                                    // тип ордера
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;     // цена для от
      request.price=NormalizeDouble(price,digits);                          // нормализова
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                               // тип ордера
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;             // цена для от
      request.price    =NormalizeDouble(price,digits);                      // нормализова
     }
   else Alert("Этот пример только для установки отложенных ордеров");       // если выбран
//--- отправка запроса
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());                     // если отправ
//--- информация об операции
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
```

© 2000-2025, MetaQuotes Ltd.

Пример торговой операции TRADE_ACTION_SLTP для изменения значений Stop Loss и Take Profit у открытой позиции:

```
#define EXPERT_MAGIC 123456  // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Модификация Stop Loss и Take Profit позиции                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление запроса и результата
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // количество открытых позиций
//--- перебор всех открытых позиций
   for(int i=0; i<total; i++)
     {
      //--- параметры ордера
      ulong  position_ticket=PositionGetTicket(i);// тикет позиции
      string position_symbol=PositionGetString(POSITION_SYMBOL); // символ
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // количест
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // MagicNumber позиции
      double volume=PositionGetDouble(POSITION_VOLUME);    // объем позиции
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss позиции
      double tp=PositionGetDouble(POSITION_TP);  // Take Profit позиции
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- вывод информации о позиции
      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);
      //--- если MagicNumber совпадает, Stop Loss и Take Profit не заданы
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- вычисление текущих ценовых уровней
         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_L
         double price_level;
         //--- если уровень минимально допустимого отступа в пунктах от текущей цены з
         if(stop_level<=0)
            stop_level=150; // зададим отступ в 150 пунктов от текущей цены закрытия
         else
            stop_level+=50; // уровень отступа возьмем равным (SYMBOL_TRADE_STOPS_LEVE

         //--- вычисление и округление значений Stop Loss и 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);
           }
         //--- обнуление значений запроса и результата
         ZeroMemory(request);
         ZeroMemory(result);
         //--- установка параметров операции
         request.action  =TRADE_ACTION_SLTP; // тип торговой операции
         request.position=position_ticket;   // тикет позиции
         request.symbol=position_symbol;     // символ
         request.sl      =sl;                // Stop Loss позиции
         request.tp      =tp;                // Take Profit позиции
         request.magic=EXPERT_MAGIC;         // MagicNumber позиции
         //--- вывод информации о модификации
         PrintFormat("Modify #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- отправка запроса
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // если отправить запро
         //--- информация об операции
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
```

Пример торговой операции TRADE_ACTION_MODIFY для модификации уровней цен отложенных ордеров:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Модификация отложенных ордеров                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // количество установленных отложенных ордеров
//--- перебор всех установленных отложенных ордеров
   for(int i=0; i<total; i++)
     {
      //--- параметры ордера
      ulong  order_ticket=OrderGetTicket(i);                                  // тикет орде
      string order_symbol=Symbol();                                           // символ
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS);       // количество
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                              // MagicNumbe
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);                     // текущий об
      double sl=OrderGetDouble(ORDER_SL);                                     // текущий St
      double tp=OrderGetDouble(ORDER_TP);                                     // текущий Ta
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE);      // тип ордера
      int offset = 50;                                                        // отступ от
      double price;                                                           // цена сраба
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);               // размер пун
      //--- вывод информации об ордере
      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);
      //--- если MagicNumber совпадает, Stop Loss и Take Profit не заданы
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                                  // тип торговой
         request.order = OrderGetTicket(i);                                   // тикет ордера
         request.symbol   =Symbol();                                          // символ
         request.deviation=5;                                                 // допустимое о
        //--- установка уровня цены, тейк-профит и стоп-лосс ордера в зависимости от е
         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);                  // нормали
           }
         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);                   // нормал
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

© 2000-2025, MetaQuotes Ltd.

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // нормал
           }
         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);                  // нормал
           }
         //--- отправка запроса
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // если отправить запро
         //--- информация об операции
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- обнуление значений запроса и результата
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
```

Пример торговой операции TRADE_ACTION_REMOVE для удаления отложенных ордеров:

```
#define EXPERT_MAGIC 123456  // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Удаление отложенных ордеров                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // количество установленных отложенных ордеров
//--- перебор всех установленных отложенных ордеров
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // тикет ордера
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // MagicNumber ордера
      //--- если MagicNumber совпадает
      if(magic==EXPERT_MAGIC)
        {
         //--- обнуление значений запроса и результата
         ZeroMemory(request);
         ZeroMemory(result);
         //--- установка параметров операции
         request.action=TRADE_ACTION_REMOVE;                   // тип торговой операци
         request.order = order_ticket;                         // тикет ордера
         //--- отправка запроса
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // если отправить запро
         //--- информация об операции
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
```

Пример торговой операции TRADE_ACTION_CLOSE_BY для закрытия позиций встречными:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Закрытие всех позиций встречными                                 |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление запроса и результата
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // количество открытых позиций
//--- перебор всех открытых позиций
   for(int i=total-1; i>=0; i--)
     {
      //--- параметры ордера
      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);
      //--- вывод информации о позиции
      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);
      //--- если MagicNumber совпадает
      if(magic==EXPERT_MAGIC)
        {
         for(int j=0; j<i; j++)
           {
            string symbol=PositionGetSymbol(j); // символ новой позиции
            //--- если символы новой и искомой позиций совпадают
            if(symbol==position_symbol && PositionGetInteger(POSITION_MAGIC)==EXPERT_M
              {
               //--- установка типа встречной позиции
               ENUM_POSITION_TYPE type_by=(ENUM_POSITION_TYPE)PositionGetInteger(POSIT
               //--- выход, если типы исходной и встречной позиций совпадают
               if(type==type_by)
                  continue;
               //--- обнуление значений запроса и результата
               ZeroMemory(request);
               ZeroMemory(result);
               //--- установка параметров операции
               request.action=TRADE_ACTION_CLOSE_BY;                    // тип то
               request.position=position_ticket;                        // тикет
               request.position_by=PositionGetInteger(POSITION_TICKET); // тикет
               //request.symbol     =position_symbol;
               request.magic=EXPERT_MAGIC;                              // MagicN
               //--- вывод информации о закрытии встречной
               PrintFormat("Close #%I64d %s %s by #%I64d",position_ticket,position_sym
               //--- отправка запроса
               if(!OrderSend(request,result))
                  PrintFormat("OrderSend error %d",GetLastError()); // если отправить
```

© 2000-2025, MetaQuotes Ltd.

```
                //--- информация об операции
                PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result
               }
            }
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

Типы торговых транзакций

В результате выполнения определенных действий с торговым счетом, его состояние изменяется. К таким действиям относятся:

- Отсылка торгового запроса любым MQL5-приложением в клиентском терминале при помощи функций OrderSend и OrderSendAsync и его последующее исполнение;
- Отсылка торгового запроса через графический интерфейс терминала и его последующее исполнение;
- Срабатывание отложенных ордеров и стоп-ордеров на сервере;
- Выполнение операций на стороне торгового сервера.

В результате данных действий, для счета выполняются торговые транзакции:

- обработка торгового запроса;
- изменение открытых ордеров;
- изменение истории ордеров;
- изменение истории сделок;
- изменение позиций.

Например, при отсылке рыночного ордера на покупку, он обрабатывается, для счета создается соответствующий ордер на покупку, происходит исполнение ордера, его удаление из списка открытых, добавление в историю ордеров, далее добавляется соответствующая сделка в историю и создается новая позиция. Все эти действия являются торговыми транзакциями.

Для того чтобы программист мог отслеживать действия, осуществляемые относительно торгового счета, предусмотрена функция OnTradeTransaction. При помощи данного обработчика в MQL5-приложении можно получать торговые транзакции, примененные к счету. Описание торговой транзакции передается в первом параметре OnTradeTransaction при помощи структуры MqlTradeTransaction.

Тип торговой транзакции передается в параметре type структуры MqlTradeTransaction. Возможные типы торговых транзакций описываются следующим перечислением:

ENUM_TRADE_TRANSACTION_TYPE

Идентификатор	Описание
TRADE_TRANSACTION_ORDER_ADD	Добавление нового открытого ордера.
TRADE_TRANSACTION_ORDER_UPDATE	Изменение открытого ордера. К данным изменениям относятся не только явные изменения со стороны клиентского терминала или торгового сервера, но также и изменение его состояния при выставлении (например, переход из состояния ORDER_STATE_STARTED в ORDER_STATE_PLACED или из ORDER_STATE_PLACED в ORDER_STATE_PARTIAL и т.д.).
TRADE_TRANSACTION_ORDER_DELETE	Удаление ордера из списка открытых. Ордер может быть удален из открытых в результате выставления

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
	соответствующего запроса либо в результате исполнения (заливки) и переноса в историю.
TRADE_TRANSACTION_DEAL_ADD	Добавление сделки в историю. Осуществляется в результате исполнения ордера или проведения операций с балансом счета.
TRADE_TRANSACTION_DEAL_UPDATE	Изменение сделки в истории. Возможны ситуации, когда ранее исполненная сделка изменяется на сервере. Например, сделка была изменена во внешней торговой системе (бирже), куда она была выведена брокером.
TRADE_TRANSACTION_DEAL_DELETE	Удаление сделки из истории. Возможны ситуации, когда ранее исполненная сделка удаляется на сервере. Например, сделка была удалена во внешней торговой системе (бирже), куда она была выведена брокером.
TRADE_TRANSACTION_HISTORY_ADD	Добавление ордера в историю в результате исполнения или отмены.
TRADE_TRANSACTION_HISTORY_UPDATE	Изменение ордера, находящегося в истории ордеров. Данный тип предусмотрен для расширения функциональности на стороне торгового сервера.
TRADE_TRANSACTION_HISTORY_DELETE	Удаление ордера из истории ордеров. Данный тип предусмотрен для расширения функциональности на стороне торгового сервера.
TRADE_TRANSACTION_POSITION	Изменение позиции, не связанное с исполнением сделки. Данный тип транзакции свидетельствует именно о том, что позиция была изменена на стороне торгового сервера. У позиции может быть изменен объем, цена открытия, а также уровни Stop Loss и Take Profit. Информация об изменениях передается в структуре MqlTradeTransaction через обработчик OnTradeTransaction. Изменение позиции (добавление, изменение или ликвидация) в результате совершения сделки не влечет за собой появление транзакции TRADE_TRANSACTION_POSITION.
TRADE_TRANSACTION_REQUEST	Уведомление о том, что торговый запрос обработан сервером, и результат его обработки получен. Для транзакций данного типа в структуре MqlTradeTransaction необходимо анализировать только одно поле - type (тип транзакции). Для получения дополнительной информации необходимо анализировать второй и третий параметры функции OnTradeTransaction (request и result).

© 2000-2025, MetaQuotes Ltd.

В зависимости от типа торговой транзакции, в структуре MqlTradeTransaction, описывающей ее, заполняются различные параметры. Подробное описание передаваемых данных приведено в разделе "Структура торговой транзакции".

Смотри также

Структура торговой транзакции, OnTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Виды заявок в стакане цен

Для биржевых инструментов доступно окно "Стакан цен", в котором можно посмотреть текущие заявки на покупку и продажу. Для каждой заявки указано желаемое направление торговой операции, требуемый объем и запрашиваемая цена.

Для получения информации о текущем состоянии стакана цен средствами языка MQL5 предназначена функция MarketBookGet(), которая помещает "снимок стакана" в массив структур MqlBookInfo. Каждый элемент этого массива в поле *type* содержит информацию о направлении заявки – это значение из перечисления ENUM_BOOK_TYPE.

ENUM_BOOK_TYPE

Идентификатор	Описание
BOOK_TYPE_SELL	Заявка на продажу
BOOK_TYPE_BUY	Заявка на покупку
BOOK_TYPE_SELL_MARKET	Заявка на продажу по рыночной цене
BOOK_TYPE_BUY_MARKET	Заявка на покупку по рыночной цене

Смотри также

Структуры и классы, Структура стакана цен, Типы торговых операций, Получение рыночной информации

© 2000-2025, MetaQuotes Ltd.

Свойства сигналов

Значения перечислений для работы с торговыми сигналами и настройками их копирования.

Перечисления свойств типа double торговых сигналов:

ENUM_SIGNAL_BASE_DOUBLE

Константа	Описание
SIGNAL_BASE_BALANCE	Баланс счета
SIGNAL_BASE_EQUITY	Средства на счете
SIGNAL_BASE_GAIN	Прирост счета в процентах
SIGNAL_BASE_MAX_DRAWDOWN	Максимальная просадка
SIGNAL_BASE_PRICE	Цена подписки на сигнал
SIGNAL_BASE_ROI	Значение ROI (Return on Investment) сигнала в %

Перечисления свойств типа integer торговых сигналов:

ENUM_SIGNAL_BASE_INTEGER

Константа	Описание
SIGNAL_BASE_DATE_PUBLISHED	Дата публикации сигнала (когда стал доступен для подписки)
SIGNAL_BASE_DATE_STARTED	Дата начала мониторинга сигнала
SIGNAL_BASE_DATE_UPDATED	Дата последнего обновления торговой статистики сигнала
SIGNAL_BASE_ID	ID сигнала
SIGNAL_BASE_LEVERAGE	Плечо торгового счета
SIGNAL_BASE_PIPS	Результат торговли в пипсах
SIGNAL_BASE_RATING	Позиция в рейтинге сигналов
SIGNAL_BASE_SUBSCRIBERS	Количество подписчиков
SIGNAL_BASE_TRADES	Количество трейдов
SIGNAL_BASE_TRADE_MODE	Тип счета (0-реальный счет, 1-демо-счет, 2-конкурсный счет)

Перечисления свойств типа string торговых сигналов:

ENUM_SIGNAL_BASE_STRING

Константа	Описание
SIGNAL_BASE_AUTHOR_LOGIN	Логин автора сигнала

© 2000-2025, MetaQuotes Ltd.

Константа	Описание
SIGNAL_BASE_BROKER	Наименование брокера (компании)
SIGNAL_BASE_BROKER_SERVER	Сервер брокера
SIGNAL_BASE_NAME	Имя сигнала
SIGNAL_BASE_CURRENCY	Валюта счета сигнала

Перечисления свойств типа double настроек копирования торговых сигналов:

ENUM_SIGNAL_INFO_DOUBLE

Константа	Описание
SIGNAL_INFO_EQUITY_LIMIT	Процент для конвертации объема сделки
SIGNAL_INFO_SLIPPAGE	Величина проскальзывания, с которым выставляются рыночные ордера при синхронизации позиций и копировании сделок
SIGNAL_INFO_VOLUME_PERCENT	Значение ограничения по средствам для сигнала, r/o

Перечисления свойств типа integer настроек копирования торговых сигналов:

ENUM_SIGNAL_INFO_INTEGER

Константа	Описание
SIGNAL_INFO_CONFIRMATIONS_DISABLED	Флаг разрешения синхронизации без показа диалога подтверждения
SIGNAL_INFO_COPY_SLTP	Флаг копирования Stop Loss и Take Profit
SIGNAL_INFO_DEPOSIT_PERCENT	Ограничения по депозиту (в %)
SIGNAL_INFO_ID	id сигнала, r/o
SIGNAL_INFO_SUBSCRIPTION_ENABLED	Флаг разрешения на копирование сделок по подписке
SIGNAL_INFO_TERMS_AGREE	Флаг согласия с условиями использования сервиса "Сигналы", r/o

Перечисления свойств типа string настроек копирования торговых сигналов:

ENUM_SIGNAL_INFO_STRING

Константа	Описание
SIGNAL_INFO_NAME	Имя сигнала, r/o

Смотри также

Управление сигналами

© 2000-2025, MetaQuotes Ltd.

Именованные константы

Все используемые в языке MQL5 константы можно разбить на следующие группы:

- Предопределенные макроподстановки - значения подставляются на этапе компиляции;
- Математические константы - значения некоторых математических выражений;
- Константы числовых типов - ограничения, накладываемые на простые типы;
- Причины деинициализации - описание причин деинициализации;
- Проверка указателя объекта - перечисление типов указателей, возвращаемых функцией CheckPointer() ;
- Прочие константы - все, что не вошло в остальные группы констант.

© 2000-2025, MetaQuotes Ltd.

Предопределенные макроподстановки

Для облегчения отладки и получения информации о работе mql5-программы введены специальные константы-макросы, значения которых устанавливается в момент компиляции. Самый простой путь использования этих констант – вывод значений с помощью функции Print(), как показано в примере.

Константа	Описание
__CPU_ARCHITECTURE__	Имя архитектуры (набор команд), под которую собран файл EX5 при компиляции
__DATE__	Дата компиляции файла без времени (часы, минуты и секунды равны 0)
__DATETIME__	Дата и время компиляции файла
__LINE__	Номер строки в исходном коде, на которой расположен данный макрос
__FILE__	Имя текущего компилируемого файла
__PATH__	Абсолютный путь к текущему компилируемому файлу
__FUNCTION__	Имя функции, в теле которой расположен макрос
__FUNCSIG__	Сигнатура функции, в теле которой расположен макрос. Вывод в лог полного описания функции с типами параметров может пригодиться при идентификации перегруженных функций
__MQLBUILD__, __MQL5BUILD__	Номер билда компилятора
__COUNTER__	Компилятор для каждого встретившегося объявления __COUNTER__ подставляет по месту значение счетчика от 0 до N-1, где N – число использований в коде. При перекомпиляции исходного кода без изменений порядок __COUNTER__ гарантируется. Значение счетчика __COUNTER__ вычисляется следующим образом: • начальное значение счетчика равно 0, • после каждого использования счетчика его значение увеличивается на 1, • сначала компилятор разворачивает все макросы и шаблоны в исходный код по месту, • для каждой специализации шаблонной функции создается отдельный код, • для каждой специализации шаблонного класса/структуры создается отдельный код, • затем компилятор проходит по полученному исходному коду в установленном порядке и заменяет каждое встретившееся использование __COUNTER__ на текущее значение счетчика.

© 2000-2025, MetaQuotes Ltd.

Константа	Описание
	В примере ниже можно наглядно увидеть, как компилятор обрабатывает исходный код и заменяет встретившиеся __COUNTER__ на последовательно возрастающие значения.
__RANDOM__	Компилятор для каждого объявления __RANDOM__ подставляет в код случайное ulong-число.

Пример:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- пример вывода информации при инициализации советника
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//--- установка интервала между событиями таймера
   EventSetTimer(5);
//---
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- пример вывода информации при деинициализации советника
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {

//--- вывод информации при поступлении тика
   Print(" __MQLBUILD__ = ",__MQLBUILD__,"  __FILE__ = ",__FILE__);
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
   test2();
//---
  }
//+------------------------------------------------------------------+
//| test1                                                            |
//+------------------------------------------------------------------+
void test1(string par)
  {
//--- вывод информации внутри функции
```

© 2000-2025, MetaQuotes Ltd.

```
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__," par=",par);
  }
//+------------------------------------------------------------------+
//| test2                                                            |
//+------------------------------------------------------------------+
void test2()
  {
//--- вывод информации внутри функции
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
  }
//+------------------------------------------------------------------+
//| OnTimer event handler                                            |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
  }
```

Пример для изучения работы с макросом __COUNTER__

```
//--- создадим макрос для быстрого вывода в журнал выражения и его значения
#define print(expr) Print(#expr,"=",expr)

//--- определим пользовательский макрос MACRO_COUNTER через предопределенный макрос __
#define MACRO_COUNTER __COUNTER__

//--- задаем значение input переменной макросом __COUNTER__
input int InpVariable = __COUNTER__;

//--- задаем значение глобальной переменной макросом __COUNTER__ перед определением фу
int ExtVariable = __COUNTER__;

//+------------------------------------------------------------------+
//| функция возвращает значение __COUNTER__                          |
//+------------------------------------------------------------------+
int GlobalFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| шаблонная функция возвращает значение __COUNTER__                |
//+------------------------------------------------------------------+
template<typename T>
int GlobalTemplateFunc(void)
  {
   return(__COUNTER__);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| структура с методом, который возвращает __COUNTER__             |
//+------------------------------------------------------------------+
struct A
  {
   int               dummy;  // не используется

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| шаблонная структура с методом, который возвращает __COUNTER__    |
//+------------------------------------------------------------------+
template<typename T>
struct B
  {
   int               dummy;  // не используется

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| структура с шаблонным методом, который возвращает __COUNTER__    |
//+------------------------------------------------------------------+
struct C
  {
   int               dummy;  // не используется

   template<typename T>
   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| функция #2, которая возвращает значение __COUNTER__              |
//+------------------------------------------------------------------+
int GlobalFunc2(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart(void)
  {
// __COUNTER__ в макросе и переменных
   print(MACRO_COUNTER);
   print(InpVariable);
   print(ExtVariable);

//--- __COUNTER__ в функциях
   print(GlobalFunc());
   print(GlobalFunc());                  // значение не меняется
   print(GlobalTemplateFunc<int>());
   print(GlobalTemplateFunc<int>());     // значение не меняется
   print(GlobalTemplateFunc<double>());// значение поменялось
   print(GlobalFunc2());
   print(GlobalFunc2());                 // значение не меняется

// __COUNTER__ в структуре
   A a1, a2;
   print(a1.Method());
   print(a2.Method());                   // значение не меняется

// __COUNTER__ в шаблонной структуре
   B<int> b1, b2;
   B<double> b3;
   print(b1.Method());
   print(b2.Method());                   // значение не меняется
   print(b3.Method());                   // значение поменялось

// __COUNTER__ в структуре с шаблонной функцией
   C c1, c2;
   print(c1.Method<int>());
   print(c1.Method<double>());           // значение поменялось
   print(c2.Method<int>());              // значение как при первом вызове c1.Method<int

//--- еще раз посмотрим __COUNTER__ в макросе и глобальной переменной
   print(MACRO_COUNTER);  // значение поменялось
   print(ExtGlobal2);
  }
//--- задаем значение глобальной переменной макросом __COUNTER__ после определений фу
int ExtGlobal2 = __COUNTER__;
//+------------------------------------------------------------------+

/* Результат
   __COUNTER__=3
   InpVariable=0
   ExtVariable=1
   GlobalFunc()=5
   GlobalFunc()=5
   GlobalTemplateFunc<int>()=8
```

© 2000-2025, MetaQuotes Ltd.

```
   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

*/
```

© 2000-2025, MetaQuotes Ltd.

Математические константы

Для некоторых математических выражений зарезервированы специальные константы, содержащие значения. Эти константы можно использовать в любом месте mql5-программы вместо вычисления их значения с помощью математических функций.

Константа	Описание	Значение
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.43429448190325182765 1
M_LN2	ln(2)	0.69314718055994530941 7
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.78539816339744830962
M_1_PI	1/pi	0.31830988618379067153 8
M_2_PI	2/pi	0.63661977236758134307 6
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.70710678118654752440 1

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- выведем значения констант
   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));
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Константы числовых типов данных

Каждый простой числовой тип предназначен для определенного круга задач и позволяет оптимизировать работу mql5-программы при правильном применении. Для лучшей читаемости кода и правильной обработки результатов вычислений введены константы, которые позволяют получить информацию об ограничениях, накладываемых на тот или иной тип простых данных.

Константа	Описание	Значение
CHAR_MIN	Минимальное значение, которое может быть представлено типом char	-128
CHAR_MAX	Максимальное значение, которое может быть представлено типом char	127
UCHAR_MAX	Максимальное значение, которое может быть представлено типом uchar	255
SHORT_MIN	Минимальное значение, которое может быть представлено типом short	-32768
SHORT_MAX	Максимальное значение, которое может быть представлено типом short	32767
USHORT_MAX	Максимальное значение, которое может быть представлено типом ushort	65535
INT_MIN	Минимальное значение, которое может быть представлено типом int	-2147483648
INT_MAX	Максимальное значение, которое может быть представлено типом int	2147483647
UINT_MAX	Максимальное значение, которое может быть представлено типом uint	4294967295
LONG_MIN	Минимальное значение, которое может быть представлено типом long	-9223372036854775808
LONG_MAX	Максимальное значение, которое может быть представлено типом long	9223372036854775807
ULONG_MAX	Максимальное значение, которое может быть представлено типом ulong	18446744073709551615

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Значение
DBL_MIN	Минимальное положительное значение, которое может быть представлено типом double	2.2250738585072014e-308
DBL_MAX	Максимальное значение, которое может быть представлено типом double	1.7976931348623158e+308
DBL_EPSILON	Наименьшее число для которого выполняется условие 1.0+DBL_EPSILON != 1.0	2.2204460492503131e-016
DBL_DIG	Количество значимых десятичных знаков	15
DBL_MANT_DIG	Количество битов в мантиссе	53
DBL_MAX_10_EXP	Максимальное десятичное значение степени экспоненты	308
DBL_MAX_EXP	Максимальное двоичное значение степени экспоненты	1024
DBL_MIN_10_EXP	Минимальное десятичное значение степени экспоненты	(-307)
DBL_MIN_EXP	Минимальное двоичное значение степени экспоненты	(-1021)
FLT_MIN	Минимальное положительное значение, которое может быть представлено типом float	1.175494351e-38
FLT_MAX	Максимальное значение, которое может быть представлено типом float	3.402823466e+38
FLT_EPSILON	Наименьшее число для которого выполняется условие 1.0+FLT_EPSILON != 1.0	1.192092896e-07
FLT_DIG	Количество значимых десятичных знаков	6
FLT_MANT_DIG	Количество битов в мантиссе	24
FLT_MAX_10_EXP	Максимальное десятичное значение степени экспоненты	38
FLT_MAX_EXP	Максимальное двоичное значение степени экспоненты	128
FLT_MIN_10_EXP	Минимальное десятичное значение степени экспоненты	-37

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Значение
FLT_MIN_EXP	Минимальное двоичное значение степени экспоненты	(-125)

Пример:

```
void OnStart()
  {
//--- выведем значения констант
   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);
   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
```

© 2000-2025, MetaQuotes Ltd.

```
   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
*/
  }
```

© 2000-2025, MetaQuotes Ltd.

Причины деинициализации

Коды причины деинициализации эксперта, возвращаемые функцией UninitializeReason(). Могут иметь любые из следующих значений:

Константа	Значение	Описание
REASON_PROGRAM	0	Эксперт прекратил свою работу, вызвав функцию ExpertRemove()
REASON_REMOVE	1	Программа удалена с графика
REASON_RECOMPILE	2	Программа перекомпилирована
REASON_CHARTCHANGE	3	Символ или период графика был изменен
REASON_CHARTCLOSE	4	График закрыт
REASON_PARAMETERS	5	Входные параметры были изменены пользователем
REASON_ACCOUNT	6	Активирован другой счет либо произошло переподключение к торговому серверу вследствие изменения настроек счета
REASON_TEMPLATE	7	Применен другой шаблон графика
REASON_INITFAILED	8	Признак того, что обработчик OnInit() вернул ненулевое значение
REASON_CLOSE	9	Терминал был закрыт

Код причины деинициализации передается также в качестве параметра предопределенной функции OnDeinit(const int reason).

Пример:

```
//+------------------------------------------------------------------+
//| get text description                                             |
//+------------------------------------------------------------------+
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:
```

© 2000-2025, 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;
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Первый способ получить код причины деинициализации
   Print(__FUNCTION__,"_Код причины деинициализации = ",reason);
//--- Второй способ получить код причины деинициализации
   Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
  }
```

© 2000-2025, MetaQuotes Ltd.

Проверка указателя объекта

Для проверки типа указателя объекта предназначена функция CheckPointer(), которая возвращает значение из перечисления ENUM_POINTER_TYPE. В случае использования некорректного указателя исполнение программы будет немедленно прервано.

Объекты, созданные оператором new, имеют тип POINTER_DYNAMIC. Только для таких указателей можно и нужно использовать оператор уничтожения delete().

Все остальные указатели имеют тип POINTER_AUTOMATIC, что означает, что данный объект был создан автоматически средой исполнения mql5-программы. Такие объекты уничтожаются после использования также автоматически.

ENUM_POINTER_TYPE

Константа	Описание
POINTER_INVALID	Некорректный указатель
POINTER_DYNAMIC	Указатель объекта, созданного оператором new
POINTER_AUTOMATIC	Указатель любого объекта, созданного автоматически (без использования new())

Смотри также

Ошибки выполнения, Оператор уничтожения объекта delete, CheckPointer()

© 2000-2025, MetaQuotes Ltd.

Прочие константы

Константа CLR_NONE служит для указания отсутствия цвета, то есть графический объект или графическая серия индикатора не будут отображены. Эта константа не вошла в список констант с наименованиями Web-цветов, но может применяться везде, где требуется указание цвета.

Константа INVALID_HANDLE может использоваться при проверке файловых хэндлов (см. функции FileOpen() и FileFindFirst()).

Константа	Описание	Значение
CHARTS_MAX	Максимально возможное количество одновременно открытых графиков в терминале	100
clrNONE	Отсутствие цвета	-1
EMPTY_VALUE	Пустое значение в индикаторном буфере	DBL_MAX
INVALID_HANDLE	Некорректный хэндл	-1
IS_DEBUG_MODE	Признак работы mq5-программы в режиме отладки	в режиме отладки не равно нулю, в противном случае 0
IS_PROFILE_MODE	Признак работы mq5-программы в режиме профилирования	в режиме профилирования не равно нулю, в противном случае 0
NULL	Ноль любого типа	0
WHOLE_ARRAY	Означает количество элементов, оставшееся до конца массива, то есть, будет обработан весь массив	-1
WRONG_VALUE	Константа может неявно приводиться к типу любого перечисления.	-1

Константа EMPTY_VALUE обычно соответствует тем значениям индикаторов, которые не отрисовываются на графике. Например, для встроенного индикатора Standard Deviation с периодом 20 не выводится на график линия для первых в истории 19 баров. Если создать хэндл этого индикатора с помощью функции iStdDev() и скопировать в массив значения индикатора для этих баров через CopyBuffer(), то эти значения как раз и будут равны EMPTY_VALUE.

Можно самостоятельно указать в пользовательском индикаторе собственное пустое значение индикатора, при котором не должна производиться отрисовка на графике. Для этого используйте функция PlotIndexSetDouble() с модификатором PLOT_EMPTY_VALUE.

Константа NULL может быть присвоена переменной любого простого типа или указателю на объект структуры или класса. Присвоение NULL строковой переменной означает полную деинициализацию этой переменной.

© 2000-2025, MetaQuotes Ltd.

Константа WRONG_VALUE предназначена для тех случаев, когда требуется вернуть значение перечисления, и это должно быть неверное значение. Например, нужно сообщить, что возвращаемое значение является значением этого перечисления. Приведем в качестве иллюстрации некую функцию CheckLineStyle(), которая возвращает стиль линии для объекта, указанного по имени. Если при запросе стиля функцией ObjectGetInteger() результатом будет true, то вернется значение перечисления ENUM_LINE_STYLE, иначе возвращается WRONG_VALUE.

```
void OnStart()
  {
   if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
      printf("Error line style getting.");
  }
//+------------------------------------------------------------------+
//| возвращает стиль линии для объекта, указанного по имени          |
//+------------------------------------------------------------------+
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);
  }
```

Константа WHOLE_ARRAY предназначена для функций, которые требуют указания количества элементов в обрабатываемых массивах:

- ArrayCopy();
- ArrayMinimum();
- ArrayMaximum();
- FileReadArray();
- FileWriteArray().

Если требуется указать, что необходимо обработать все значения массива с указанной позиции и до конца, то достаточно указать значение WHOLE_ARRAY.

Константа IS_PROFILE_MODE позволяет изменить работу программы для корректного сбора информации в режиме профилирования. Профилирование позволяет замерить время выполнения отдельных фрагментов программы (обычно это функции), а также подсчитать количество таких вызовов. Для корректного получения информации о времени выполнения в режиме профилировки можно отключить вызовы функции Sleep() как в примере:

```
//--- Sleep может сильно повлиять(исказить) на результат профилировки
if(!IS_PROFILE_MODE) Sleep(100); // запрещаем вызов Sleep() в режиме профилировки
```

Значение константы IS_PROFILE_MODE задается компилятором в момент компиляции, и в обычном режиме выставляется равным нулю. При запуске программы в режиме профилирования

© 2000-2025, MetaQuotes Ltd.

производится специальная компиляция, и в этом случае вместо IS_PROFILE_MODE подставляется значение отличное от нуля.

Константа IS_DEBUG_MODE пригодится в тех случаях, когда необходимо немного изменить работу mql5-программы в режиме отладки. Например, в режиме отладки может потребоваться выводить дополнительную отладочную информацию в лог терминала или создавать вспомогательные графические объекты на графике.

Приведенный ниже пример создает объект Label и задает ее описание и цвет в зависимости от того, в каком режиме выполняется скрипт. Для того чтобы запустить скрипт в режиме отладки из MetaEditor, нажмите клавишу F5. Если запустить скрипта из окна навигатора в терминале, то цвет и текст объекта Label будут другими.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="invisible_label";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object ",label_name," not found. Error code = ",GetLastError());
      //--- создадим объект Label
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- установим координату X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- установим координату Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      ResetLastError();
      if(IS_DEBUG_MODE) // режим отладки
        {
         //--- выведем сообщение о режиме выполнения скрипта
         ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
         //--- зададим красный цвет текста
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
            Print("Не удалось установить цвет. Ошибка ",GetLastError());
        }
      else               // рабочий режим
        {
         ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- зададим невидимый цвет текста
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("Не удалось установить цвет. Ошибка ",GetLastError());
        }
      ChartRedraw();
      DebugBreak();     // здесь произойдет прерывание, если мы в режиме отладки
     }
  }
```

Методы шифрования данных

Для указания метода преобразования данных (шифрование и расчет хешей) в функциях CryptEncode() и CryptDecode() используется перечисление ENUM_CRYPT_METHOD.

ENUM_CRYPT_METHOD

Константа	Описание
CRYPT_BASE64	Шифрование BASE64 (перекодировка)
CRYPT_AES128	Шифрование AES с ключом 128 бит (16 байт)
CRYPT_AES256	Шифрование AES с ключом 256 бит (32 байта)
CRYPT_DES	Шифрование DES с ключом 56 бит (7 байт)
CRYPT_HASH_SHA1	Расчёт HASH SHA1
CRYPT_HASH_SHA256	Расчёт HASH SHA256
CRYPT_HASH_MD5	Расчёт HASH MD5
CRYPT_ARCH_ZIP	ZIP архивирование

Смотри также

DebugBreak, Информация о запущенной MQL5-программе, CryptEncode(), CryptDecode()

© 2000-2025, MetaQuotes Ltd.

Структуры данных

В MQL5 существуют 12 предопределенных структур, предназначенные для хранения и передачи служебной информации:

- MqlDateTime предназначена для представления даты и времени;
- MqlParam позволяет передавать входные параметры при создании хэндла индикатора с помощью функции IndicatorCreate();
- MqlRates предоставляет информацию об исторических данных, содержащих цену, объем и спред;
- MqlBookInfo для получения информации, отображаемой в стакане цен (окно котировок);
- MqlTradeRequest для создания торгового запроса при проведении торговых операций;
- MqlTradeCheckResult позволяет проверить подготовленный торговый запрос перед его отправкой;
- MqlTradeResult содержит ответ торгового сервера на торговый запрос, отправленный функцией OrderSend();
- MqlTradeTransaction содержит описание торговой транзакции;
- MqlTick предназначена для быстрого получения наиболее востребованной информации о текущих ценах.
- Структуры Экономического календаря предназначены для получения информации о событиях Экономического календаря, которые поступают в платформу MetaTrader 5 в режиме реального времени. Функции Экономического календаря позволяют проводить анализ макроэкономических индикаторов сразу же после публикации новых отчетов, так как актуальные значения транслируются напрямую из источника без задержек.

© 2000-2025, MetaQuotes Ltd.

MqlDateTime

Структура даты содержит в себе восемь полей типа int.

```
struct MqlDateTime
  {
   int year;            // год
   int mon;             // месяц
   int day;             // день
   int hour;            // час
   int min;             // минуты
   int sec;             // секунды
   int day_of_week;     // день недели (0-воскресенье, 1-понедельник, ... ,6-суббота)
   int day_of_year;     // порядковый номер в году (1 января имеет номер 0)
  };
```

Примечание

Порядковый номер в году day_of_year в високосном году, начиная с марта, будет отличаться от порядкового номера соответствующего дня в невисокосном году.

Пример:

```
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);
  }
/*  Результат
   01.03.2008, day of year = 60
   01.03.2009, day of year = 59
*/
```

Смотри также

TimeToStruct, Структуры и классы

© 2000-2025, MetaQuotes Ltd.

Структура входных параметров индикатора (MqlParam)

Структура MqlParam специально разработана для передачи входных параметров при создании хэндла технического индикатора с помощью функции IndicatorCreate().

```
struct MqlParam
  {
   ENUM_DATATYPE     type;                    // тип входного параметра, значение пере
   long              integer_value;           // поле для хранения целочисленного знач
   double            double_value;            // поле для хранения значения double или
   string            string_value;            // поле для хранения значения строкового
  };
```

Все входные параметры индикатора передаются в виде массива типа MqlParam, поле *type* каждого элемента этого массива указывает тип данных, передаваемых данным элементом. Сами значения параметров индикатора необходимо предварительно поместить в соответствующие поля каждого элемента (в *integer_value*, в *double_value* или в *string_value*) в зависимости от того, какое значение перечисления ENUM_DATATYPE содержится в поле *type*.

Если функции IndicatorCreate() третьим параметром в качестве типа индикатора передается значение IND_CUSTOM, то первый элемент массива входных параметров должен иметь поле *type* со значением TYPE_STRING из перечисления ENUM_DATATYPE, а поле *string_value* должно содержать имя пользовательского индикатора.

© 2000-2025, MetaQuotes Ltd.

MqlRates

Структура для хранения информации о ценах, объемах и спреде.

```
struct MqlRates
  {
   datetime time;         // время начала периода
   double   open;         // цена открытия
   double   high;         // наивысшая цена за период
   double   low;          // наименьшая цена за период
   double   close;        // цена закрытия
   long     tick_volume;  // тиковый объем
   int      spread;       // спред
   long     real_volume;  // биржевой объем
  };
```

Пример:

```
void OnStart()
  {
   MqlRates rates[];
   int copied=CopyRates(NULL,0,0,100,rates);
   if(copied<=0)
      Print("Ошибка копирования ценовых данных ",GetLastError());
   else
      Print("Скопировано ",ArraySize(rates)," баров");
  }
```

Смотри также

CopyRates, Доступ к таймсериям

© 2000-2025, MetaQuotes Ltd.

MqlBookInfo

Структура, предоставляющая информацию в стакане цен.

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE    type;           // тип заявки из перечисления ENUM_BOOK_TYPE
   double            price;          // цена
   long              volume;         // объем
   double            volume_real;    // объем с повышенной точностью
  };
```

Примечание

Структура MqlBookInfo является предопределенной, поэтому ее объявление и описание не требуется. Чтобы использовать структуру, достаточно объявить переменную данного типа.

Стакан доступен не для всех финансовых инструментов.

Пример:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo по ",Symbol());
     }
   else
      Print("Не удалось получить содержимое стакана по символу ",Symbol());
```

Смотри также

MarketBookAdd, MarketBookRelease, MarketBookGet, Виды заявок в стакане цен, Типы данных

© 2000-2025, MetaQuotes Ltd.

Структура торгового запроса (MqlTradeRequest)

Взаимодействие клиентского терминала и торгового сервера для проведения операций постановки ордеров производится посредством торговых запросов. Запрос представлен специальной предопределенной структурой MqlTradeRequest, которая содержит все поля, необходимые для заключения торговых сделок. Результат обработки запроса представлен структурой MqlTradeResult.

```
struct MqlTradeRequest
  {
   ENUM_TRADE_REQUEST_ACTIONS    action;           // Тип выполняемого действия
   ulong                         magic;            // Штамп эксперта (идентификатор ма
   ulong                         order;            // Тикет ордера
   string                        symbol;           // Имя торгового инструмента
   double                        volume;           // Запрашиваемый объем сделки в лот
   double                        price;            // Цена
   double                        stoplimit;        // Уровень StopLimit ордера
   double                        sl;               // Уровень Stop Loss ордера
   double                        tp;               // Уровень Take Profit ордера
   ulong                         deviation;        // Максимально приемлемое отклонени
   ENUM_ORDER_TYPE               type;             // Тип ордера
   ENUM_ORDER_TYPE_FILLING       type_filling;     // Тип ордера по исполнению
   ENUM_ORDER_TYPE_TIME          type_time;        // Тип ордера по времени действия
   datetime                      expiration;       // Срок истечения ордера (для ордер
   string                        comment;          // Комментарий к ордеру
   ulong                         position;         // Тикет позиции
   ulong                         position_by;      // Тикет встречной позиции
  };
```

Описание полей

Поле	Описание
action	Тип торговой операции. Значение может быть одним из значений перечисления ENUM_TRADE_REQUEST_ACTIONS
magic	Идентификатор эксперта. Позволяет организовать аналитическую обработку торговых ордеров. Каждый эксперт может выставлять свой собственный уникальный идентификатор при отправке торгового запроса
order	Тикет ордера. Требуется для модификации отложенных ордеров
symbol	Имя торгового инструмента, по которому выставляется ордер. Не требуется при операциях модификации ордеров и закрытии позиций
volume	Запрашиваемый объем сделки в лотах. Реальное значение объема при открытии сделки будет зависеть от типа ордера по исполнению.
price	Цена, при достижении которой ордер должен быть исполнен. Для рыночных ордеров по инструментам с типом исполнения "Market

© 2000-2025, MetaQuotes Ltd.

Поле	Описание
	Execution" (SYMBOL_TRADE_EXECUTION_MARKET), имеющих тип TRADE_ACTION_DEAL, указание цены не требуется
stoplimit	Цена, по которой будет выставлен отложенный Limit ордер, при достижении ценой значения price (это условие является обязательным). До этого момента отложенный ордер в торговую систему не выводится
sl	Цена, по которой сработает Stop Loss ордер при движении цены в неблагоприятном направлении
tp	Цена, по которой сработает Take Profit ордер при движении цены в благоприятном направлении
deviation	Максимально приемлемое отклонение от запрашиваемой цены, задаваемое в пунктах
type	Тип ордера. Значение может быть одним из значений перечисления ENUM_ORDER_TYPE
type_filling	Тип ордера по исполнению. Значение может быть одним из значений ENUM_ORDER_TYPE_FILLING
type_time	Тип ордера по по истечению. Значение может быть одним из значений ENUM_ORDER_TYPE_TIME
expiration	Срок истечения отложенного ордера (для ордеров типа ORDER_TIME_SPECIFIED)
comment	Комментарий к ордеру
position	Тикет позиции. Следует заполнять при изменении и закрытии позиции для ее однозначной идентификации. Как правило, соответствует тикету ордера, в результате которого позиция была открыта.
position_by	Тикет встречной позиции. Используется при закрытии позиции встречной — открытой по тому же инструменту, но в противоположном направлении.

При модификации или закрытии позиции в системе хеджинга обязательно указывайте ее тикет (MqlTradeRequest::position). В системе неттинга тикет также можно указывать, однако идентификации позиции осуществляется по имени символа.

Для отправки приказов на совершение торговых операций необходимо использовать функцию OrderSend(). Для каждой торговой операции необходимо указывать обязательные поля и можно заполнять опциональные поля. Всего предусмотрено семь вариантов отправки торгового запроса:

Request Execution

Торговый ордер на открытие позиции в режиме Request Execution (режим торговли по запросу текущих цен). Требуется указание 9 полей:

- action
- symbol
- volume

© 2000-2025, MetaQuotes Ltd.

- price
- sl
- tp
- deviation
- type
- type_filling

Можно также задать значения полей magic и comment.

Instant Execution

Торговый ордер на открытие позиции в режиме Instant Execution (режим торговли по потоковым ценам). Требуется указание 9 полей:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

Можно также задать значения полей magic и comment.

Market Execution

Торговый ордер на открытие позиции в режиме Market Execution (режим исполнения торговых приказов по рынку). Требуется указание 5 полей:

- action
- symbol
- volume
- type
- type_filling

Можно также задать значения полей magic и comment.

Exchange Execution

Торговый ордер на открытие позиции в режиме Exchange Execution (биржевой режим исполнения торговых приказов). Требуется указание 5 полей:

- action
- symbol
- volume
- type
- type_filling

Можно также задать значения полей magic и comment.

Пример торговой операции TRADE_ACTION_DEAL для открытия позиции Buy:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Открытие позиции Buy                                             |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- параметры запроса
   request.action   =TRADE_ACTION_DEAL;                    // тип торговой операции
   request.symbol   =Symbol();                             // символ
   request.volume   =0.1;                                  // объем в 0.1 лот
   request.type     =ORDER_TYPE_BUY;                       // тип ордера
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // цена для открытия
   request.deviation=5;                                    // допустимое отклонение
   request.magic    =EXPERT_MAGIC;                         // MagicNumber ордера
//--- отправка запроса
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // если отправить запрос
//--- информация об операции
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
```

Пример торговой операции TRADE_ACTION_DEAL для открытия позиции Sell:

```
#define EXPERT_MAGIC 123456   // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Открытие позиции Sell                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- параметры запроса
   request.action   =TRADE_ACTION_DEAL;                    // тип торговой операции
   request.symbol   =Symbol();                             // символ
   request.volume   =0.2;                                  // объем в 0.2 лот
   request.type     =ORDER_TYPE_SELL;                      // тип ордера
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // цена для открытия
   request.deviation=5;                                    // допустимое отклонение
   request.magic    =EXPERT_MAGIC;                         // MagicNumber ордера
//--- отправка запроса
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // если отправить запрос
//--- информация об операции
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
```

Пример торговой операции TRADE_ACTION_DEAL для закрытия позиций:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Закрытие всех позиций                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление запроса и результата
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // количество открытых позиций
//--- перебор всех открытых позиций
   for(int i=total-1; i>=0; i--)
     {
      //--- параметры ордера
      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);
      //--- вывод информации о позиции
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- если MagicNumber совпадает
      if(magic==EXPERT_MAGIC)
        {
         //--- обнуление значений запроса и результата
         ZeroMemory(request);
         ZeroMemory(result);
         //--- установка параметров операции
         request.action   =TRADE_ACTION_DEAL;        // тип торговой операции
         request.position =position_ticket;          // тикет позиции
         request.symbol   =position_symbol;          // символ
         request.volume   =volume;                   // объем позиции
         request.deviation=5;                        // допустимое отклонение от цены
         request.magic    =EXPERT_MAGIC;             // MagicNumber позиции
         //--- установка цены и типа ордера в зависимости от типа позиции
         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;
           }
         //--- вывод информации о закрытии
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToString
         //--- отправка запроса
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // если отправить запро
         //--- информация об операции
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
```

SL & TP Modification

Торговый приказ на модификацию уровней StopLoss и/или TakeProfit. Требуется указание 4 полей:

- action
- symbol
- sl
- tp
- position

Пример торговой операции TRADE_ACTION_SLTP для изменения значений Stop Loss и Take Profit у открытой позиции:

```
#define EXPERT_MAGIC 123456  // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Модификация Stop Loss и Take Profit позиции                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление запроса и результата
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // количество открытых позиций
//--- перебор всех открытых позиций
   for(int i=0; i<total; i++)
     {
      //--- параметры ордера
      ulong  position_ticket=PositionGetTicket(i);// тикет позиции
      string position_symbol=PositionGetString(POSITION_SYMBOL); // символ
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // количест
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // MagicNumber позиции
      double volume=PositionGetDouble(POSITION_VOLUME);    // объем позиции
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss позиции
      double tp=PositionGetDouble(POSITION_TP);  // Take Profit позиции
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- вывод информации о позиции
      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);
      //--- если MagicNumber совпадает, Stop Loss и Take Profit не заданы
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- вычисление текущих ценовых уровней
         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_L
         double price_level;
         //--- если уровень минимально допустимого отступа в пунктах от текущей цены з
         if(stop_level<=0)
            stop_level=150; // зададим отступ в 150 пунктов от текущей цены закрытия
         else
            stop_level+=50; // уровень отступа возьмем равным (SYMBOL_TRADE_STOPS_LEVE

         //--- вычисление и округление значений Stop Loss и 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);
           }
         //--- обнуление значений запроса и результата
         ZeroMemory(request);
         ZeroMemory(result);
         //--- установка параметров операции
         request.action  =TRADE_ACTION_SLTP; // тип торговой операции
         request.position=position_ticket;   // тикет позиции
         request.symbol=position_symbol;     // символ
         request.sl      =sl;                // Stop Loss позиции
         request.tp      =tp;                // Take Profit позиции
         request.magic=EXPERT_MAGIC;         // MagicNumber позиции
         //--- вывод информации о модификации
         PrintFormat("Modify #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- отправка запроса
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // если отправить запро
         //--- информация об операции
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
```

Pending Order

Торговый приказ на установку отложенного ордера. Требуется указание 11 полей:

- action
- symbol
- volume
- price
- stoplimit
- sl
- tp
- type
- type_filling

© 2000-2025, MetaQuotes Ltd.

- type_time
- expiration

Можно также задать значения полей magic и comment.

Пример торговой операции TRADE_ACTION_PENDING для установки отложенного ордера:

© 2000-2025, MetaQuotes Ltd.

```
#property description "Пример установки отложенных ордеров"
#property script_show_inputs
#define EXPERT_MAGIC 123456                                  // MagicNumber эксперта
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;    // тип ордера
//+------------------------------------------------------------------+
//| Установка отложенных ордеров                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- параметры для установки отложенного ордера
   request.action   =TRADE_ACTION_PENDING;                                  // тип торговой
   request.symbol   =Symbol();                                              // символ
   request.volume   =0.1;                                                   // объем в 0.1
   request.deviation=2;                                                     // допустимое о
   request.magic    =EXPERT_MAGIC;                                          // MagicNumber
   int offset = 50;                                                         // отступ от те
   double price;                                                            // цена срабать
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);                     // размер пункт
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);                     // кол-во знако
   //--- проверка типа операции
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                               // тип ордера
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;             // цена для отк
      request.price    =NormalizeDouble(price,digits);                      // нормализован
     }
   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);                       // нормализова
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                                    // тип ордера
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;     // цена для от
      request.price=NormalizeDouble(price,digits);                          // нормализова
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                                // тип ордера
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;             // цена для от
      request.price    =NormalizeDouble(price,digits);                       // нормализова
     }
   else Alert("Этот пример только для установки отложенных ордеров");       // если выбран
//--- отправка запроса
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());                     // если отправ
//--- информация об операции
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Modify Pending Order

Торговый приказ на модификацию уровней цен отложенного ордера. Требуется указание 7 полей:

© 2000-2025, MetaQuotes Ltd.

- action
- order
- price
- sl
- tp
- type_time
- expiration

Пример торговой операции TRADE_ACTION_MODIFY для модификации уровней цен отложенного ордера:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber эксперта
//+------------------------------------------------------------------+
//| Модификация отложенных ордеров                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // количество установленных отложенных ордеров
//--- перебор всех установленных отложенных ордеров
   for(int i=0; i<total; i++)
     {
      //--- параметры ордера
      ulong  order_ticket=OrderGetTicket(i);                          // тикет орде
      string order_symbol=Symbol();                                   // символ
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS); // количество
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                      // MagicNumbe
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);             // текущий об
      double sl=OrderGetDouble(ORDER_SL);                             // текущий St
      double tp=OrderGetDouble(ORDER_TP);                             // текущий Ta
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // тип ордера
      int offset = 50;                                                // отступ от
      double price;                                                   // цена сраба
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);       // размер пун
      //--- вывод информации об ордере
      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);
      //--- если MagicNumber совпадает, Stop Loss и Take Profit не заданы
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                          // тип торговой
         request.order = OrderGetTicket(i);                           // тикет ордера
         request.symbol   =Symbol();                                  // символ
         request.deviation=5;                                         // допустимое о
        //--- установка уровня цены, тейк-профит и стоп-лосс ордера в зависимости от е
         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);              // нормали
           }
         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);              // нормал
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

© 2000-2025, MetaQuotes Ltd.

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // нормал
           }
         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);                  // нормал
           }
         //--- отправка запроса
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // если отправить запро
         //--- информация об операции
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- обнуление значений запроса и результата
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

Delete Pending Order

Торговый приказ на удаление отложенного ордера. Требуется указание 2 полей:

- action
- order

Пример торговой операции TRADE_ACTION_REMOVE для удаления отложенных ордеров:

```
#define EXPERT_MAGIC 123456  // MagicNumber эксперта
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Удаление отложенных ордеров                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- объявление и инициализация запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // количество установленных отложенных ордеров
//--- перебор всех установленных отложенных ордеров
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // тикет ордера
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // MagicNumber ордера
      //--- если MagicNumber совпадает
      if(magic==EXPERT_MAGIC)
        {
         //--- обнуление значений запроса и результата
         ZeroMemory(request);
         ZeroMemory(result);
         //--- установка параметров операции
         request.action=TRADE_ACTION_REMOVE;                   // тип торговой операци
         request.order = order_ticket;                         // тикет ордера
         //--- отправка запроса
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // если отправить запро
         //--- информация об операции
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Смотри также

Структуры и классы, Торговые функции, Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

Структура результата проверки торгового запроса (MqlTradeCheckResult)

Прежде чем отправить торговому серверу запрос на торговую операцию, рекомендуется провести его проверку. Проверка осуществляется функцией OrderCheck(), которой передается сам проверяемый запрос и переменная типа структуры MqlTradeCheckResult. В эту переменную и будет записан результат проверки.

```
struct MqlTradeCheckResult
  {
   uint          retcode;             // Код ответа
   double        balance;             // Баланс после совершения сделки
   double        equity;              // Эквити после совершения сделки
   double        profit;              // Плавающая прибыль
   double        margin;              // Маржевые требования
   double        margin_free;         // Свободная маржа
   double        margin_level;        // Уровень маржи
   string        comment;             // Комментарий к коду ответа (описание ошибки)
  };
```

Описание полей

Поле	Описание
retcode	Код возврата
balance	Значение баланса, которое будет после выполнения торговой операции
equity	Значение собственных средств, которое будет после выполнения торговой операции
profit	Значение плавающей прибыли, которое будет после выполнения торговой операции
margin	Размер маржи необходимый для требуемой торговой операции
margin_free	Размер свободных собственных средств, которые останутся после выполнения требуемой торговой операции
margin_level	Уровень маржи, который установится после выполнения требуемой торговой операции
comment	Комментарий к коду ответа, описание ошибки

Смотри также

Структура торгового запроса, Структура для получения текущих цен, OrderSend, OrderCheck

© 2000-2025, MetaQuotes Ltd.

Структура результата торгового запроса (MqlTradeResult)

В ответ на торговый запрос постановки ордера в торговую систему, торговый сервер возвращает данные, содержащие информацию о результате обработки торгового запроса в виде специальной предопределенной структуры MqlTradeResult.

```
struct MqlTradeResult
  {
   uint     retcode;          // Код результата операции
   ulong    deal;             // Тикет сделки, если она совершена
   ulong    order;            // Тикет ордера, если он выставлен
   double   volume;           // Объем сделки, подтверждённый брокером
   double   price;            // Цена в сделке, подтверждённая брокером
   double   bid;              // Текущая рыночная цена предложения (цены реквота)
   double   ask;              // Текущая рыночная цена спроса (цены реквота)
   string   comment;          // Комментарий брокера к операции (по умолчанию заполняе
   uint     request_id;       // Идентификатор запроса, устанавливается терминалом при
   int      retcode_external; // Код ответа внешней торговой системы
  };
```

Описание полей

Поле	Описание
retcode	Код возврата торгового сервера
deal	Тикет сделки, если она совершена. Сообщается при торговой операции TRADE_ACTION_DEAL
order	Тикет ордера, если он выставлен. Сообщается при торговой операции TRADE_ACTION_PENDING
volume	Объем сделки, подтверждённый брокером. Зависит от типа ордера по исполнению
price	Цена в сделке, подтверждённая брокером. Зависит от поля *deviation* в торговом запросе и/или от типа торговой операции
bid	Текущая рыночная цена предложения (цены реквоты)
ask	Текущая рыночная цена спроса (цены реквоты)
comment	Комментарий брокера к операции (по умолчанию заполняется расшифровкой кода возврата торгового сервера)
request_id	Идентификатор запроса, проставляемый терминалом при отсылке на торговый сервер
retcode_external	Код ошибки, которую вернула внешняя торговая система. Проставление и виды этих ошибок зависят от брокера и внешней торговой системы, в которую выводятся торговые операции

Результат торговой операции возвращается в переменную типа MqlTradeResult, которая передается вторым параметром в функцию OrderSend() для проведения торговых операций.

© 2000-2025, MetaQuotes Ltd.

Терминал записывает идентификатор запроса в поле request_id при его отправке на торговый сервер функциями OrdersSend() и OrderSendAsync(). От торгового сервера терминал получает сообщения о совершенных торговых транзакциях и передает их на обработку в функцию OnTradeTransaction(), которая содержит в качестве параметров:

- описание самой торговой транзакции в структуре MqlTradeTransaction;
- описание торгового запроса, отправленного из функции OrderSend() или OrdersSendAsync(). Идентификатор запроса отправляется терминалом на торговый сервер, а сам запрос и его request_id сохраняются в памяти терминала;
- результат исполнения торгового запроса в виде структуры MqlTradeResult, в котором поле request_id содержит идентификатор этого самого запроса.

Функция OnTradeTransaction() получает три входных параметра, но последние два параметра имеет смысл анализировать только для торговых транзакций, имеющих тип TRADE_TRANSACTION_REQUEST. Во всех остальных случаях данные о торговом запросе и результате его выполнения не заполняются. Пример анализа параметров приведен в разделе Структура торговой транзакции.

Установка терминалом идентификатора request_id для торгового запроса при его отправке на сервер в первую очередь предназначена для работы с асинхронной функцией OrderSendAsync(). Этот идентификатор позволяет связать выполненное действие (вызов функций OrderSend или OrderSendAsync) с результатом этого действия, передаваемым в OnTradeTransaction().

Пример:

```
//+------------------------------------------------------------------+
//| Отправка торгового запроса с обработкой результата               |
//+------------------------------------------------------------------+
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
  {
//--- сбросим код последней ошибки в ноль
   ResetLastError();
//--- отправим запрос
   bool success=OrderSend(request,result);
//--- если результат неудачный - попробуем узнать в чем дело
   if(!success)
     {
      int answer=result.retcode;
      Print("TradeLog: Trade request failed. Error = ",GetLastError());
      switch(answer)
        {
         //--- реквота
         case 10004:
           {
            Print("TRADE_RETCODE_REQUOTE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- ордер не принят сервером
```

© 2000-2025, MetaQuotes Ltd.

```
         case 10006:
           {
            Print("TRADE_RETCODE_REJECT");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- неправильная цена
         case 10015:
           {
            Print("TRADE_RETCODE_INVALID_PRICE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- неправильный SL и/или TP
         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;
           }
         //--- некорректный объем
         case 10014:
           {
            Print("TRADE_RETCODE_INVALID_VOLUME");
            Print("request.volume = ",request.volume,"   result.volume = ",
                  result.volume);
            break;
           }
         //--- не хватает денег на торговую операцию
         case 10019:
           {
            Print("TRADE_RETCODE_NO_MONEY");
            Print("request.volume = ",request.volume,"   result.volume = ",
                  result.volume,"   result.comment = ",result.comment);
            break;
           }
         //--- какая-то другая причина, сообщим код ответа сервера
         default:
           {
            Print("Other answer = ",answer);
           }
        }
      //--- сообщим о неудачном результате торгового запроса возвратом false
      return(false);
     }
//--- OrderSend() вернул true - повторим ответ
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

Структура торговой транзакции (MqlTradeTransaction)

В результате выполнения определенных действий с торговым счетом, его состояние изменяется. К таким действиям относятся:

- Отсылка торгового запроса любым MQL5-приложением в клиентском терминале при помощи функций OrderSend и OrderSendAsync и его последующее исполнение;
- Отсылка торгового запроса через графический интерфейс терминала и его последующее исполнение;
- Срабатывания отложенных ордеров и стоп-ордеров на сервере;
- Выполнения операций на стороне торгового сервера.

В результате данных действий для счета выполняются торговые транзакции:

- обработка торгового запроса;
- изменение открытых ордеров;
- изменение истории ордеров;
- изменение истории сделок;
- изменение позиций.

Например, при отсылке рыночного ордера на покупку он обрабатывается, для счета создается соответствующий ордер на покупку, происходит исполнение ордера, его удаление из списка открытых, добавление в историю ордеров, далее добавляется соответствующая сделка в историю и создается новая позиция. Все эти действия являются торговыми транзакциями.

Для получения торговых транзакций, примененных к счету, в MQL5 предусмотрен специальный обработчик OnTradeTransaction(). В первый параметр этого обработчика передается структура MqlTradeTransaction, описывающая торговые транзакции.

```
struct MqlTradeTransaction
  {
   ulong                     deal;             // Тикет сделки
   ulong                     order;            // Тикет ордера
   string                    symbol;           // Имя торгового инструмента
   ENUM_TRADE_TRANSACTION_TYPE type;           // Тип торговой транзакции
   ENUM_ORDER_TYPE           order_type;       // Тип ордера
   ENUM_ORDER_STATE          order_state;      // Состояние ордера
   ENUM_DEAL_TYPE            deal_type;        // Тип сделки
   ENUM_ORDER_TYPE_TIME      time_type;        // Тип ордера по времени действия
   datetime                  time_expiration;  // Срок истечения ордера
   double                    price;            // Цена
   double                    price_trigger;    // Цена срабатывания стоп-лимитного
   double                    price_sl;         // Уровень Stop Loss
   double                    price_tp;         // Уровень Take Profit
   double                    volume;           // Объем в лотах
   ulong                     position;         // Тикет позиции
   ulong                     position_by;      // Тикет встречной позиции
  };
```

© 2000-2025, MetaQuotes Ltd.

Описание полей

Поле	Описание
deal	Тикет сделки.
order	Тикет ордера.
symbol	Имя торгового инструмента, по которому совершена транзакция.
type	Тип торговой транзакции. Значение может быть одним из значений перечисления ENUM_TRADE_TRANSACTION_TYPE.
order_type	Тип торгового ордера. Значение может быть одним из значений перечисления ENUM_ORDER_TYPE.
order_state	Состояние торгового ордера. Значение может быть одним из значений перечисления ENUM_ORDER_STATE.
deal_type	Тип сделки. Значение может быть одним из значений перечисления ENUM_DEAL_TYPE.
time_type	Тип ордера по истечению. Значение может быть одним из значений ENUM_ORDER_TYPE_TIME.
time_expiration	Срок истечения отложенного ордера (для ордеров типа ORDER_TIME_SPECIFIED и ORDER_TIME_SPECIFIED_DAY).
price	Цена. В зависимости от типа торговой транзакции может быть ценой ордера, сделки или позиции.
price_trigger	Стоп-цена (цена срабатывания) стоп-лимитного ордера (ORDER_TYPE_BUY_STOP_LIMIT и ORDER_TYPE_SELL_STOP_LIMIT).
price_sl	Цена Stop Loss. В зависимости от типа торговой транзакции может относиться к ордеру, сделке или позиции.
price_tp	Цена Take Profit. В зависимости от типа торговой транзакции может относиться к ордеру, сделке или позиции.
volume	Объем в лотах. В зависимости от типа торговой транзакции может указывать на текущий объем ордера, объем сделки или объем позиции.
position	Тикет позиции, на которую повлияла транзакция.
position_by	Тикет встречной позиции. Используется при закрытии позиции встречной — открытой по тому же инструменту, но в противоположном направлении.

Определяющим параметром для анализа поступившей транзакции является ее тип, который передается в поле **type**. Например, если транзакция является типом TRADE_TRANSACTION_REQUEST (получен результат обработки торгового запроса сервером), то структура имеет только одно заполненное поле **type**, остальные поля анализировать не нужно. В этом случае можно произвести анализ двух дополнительных параметров **request** и **result**, которые передаются в обработчик OnTradeTransaction(), как это показано в примере ниже.

Зная тип торговой операции, можно принять решение об анализе текущего состояния ордеров, позиций и сделок на торговом счете. Необходимо иметь в виду, что один торговый запрос,

© 2000-2025, MetaQuotes Ltd.

отправленный из терминала серверу, может породить несколько торговых транзакций, очередность поступления которых в терминал не гарантируется.

Структура MqlTradeTransaction заполняется по-разному в зависимости от типа торговой транзакции (ENUM_TRADE_TRANSACTION_TYPE):

TRADE_TRANSACTION_ORDER_* и TRADE_TRANSACTION_HISTORY_*

Для торговых транзакций, касающихся обработки открытых ордеров (TRADE_TRANSACTION_ORDER_ADD, TRADE_TRANSACTION_ORDER_UPDATE и TRADE_TRANSACTION_ORDER_DELETE) и истории ордеров (TRADE_TRANSACTION_HISTORY_ADD, TRADE_TRANSACTION_HISTORY_UPDATE, TRADE_TRANSACTION_HISTORY_DELETE), в структуре MqlTradeTransaction заполняются следующие поля:

- order - тикет ордера;
- symbol - имя финансового инструмента в ордере;
- type - тип торговой транзакции;
- order_type - тип ордера;
- orders_state - текущее состояние ордера;
- time_type - тип истечения ордера;
- time_expiration - время истечения ордера (для ордеров с типом истечения ORDER_TIME_SPECIFIED и ORDER_TIME_SPECIFIED_DAY);
- price - цена в ордере, указанная клиентом;
- price_trigger - стоп-цена срабатывания стоп-лимитного ордера (только для ORDER_TYPE_BUY_STOP_LIMIT и ORDER_TYPE_SELL_STOP_LIMIT);
- price_sl - цена Stop Loss ордера (заполняется, если указана в ордере);
- price_tp - цена Take Profit ордера (заполняется, если указана в ордере);
- volume - текущий объем ордера (не исполненный). Изначальный объем ордера можно узнать из истории ордеров при помощи функций HistoryOrders*.
- position - тикет позиции, открытой, измененной или закрытой в результате исполнения ордера. Заполняется только для рыночных ордеров. Не заполняется для TRADE_TRANSACTION_ORDER_ADD.
- position_by - тикет встречной позиции. Заполняется только для ордеров на закрытие позиции встречной (close by).

TRADE_TRANSACTION_DEAL_*

Для торговых транзакций, касающихся обработки сделок (TRADE_TRANSACTION_DEAL_ADD, TRADE_TRANSACTION_DEAL_UPDATE и TRADE_TRANSACTION_DEAL_DELETE), в структуре MqlTradeTransaction заполняются следующие поля:

- deal - тикет сделки;
- order - тикет ордера, на основе которого совершена сделка;
- symbol - имя финансового инструмента в сделке;
- type - тип торговой транзакции;
- deal_type - тип сделки;
- price - цена, по которой совершена сделка;
- price_sl - цена Stop Loss (заполняется, если указана в ордере, на основе которого совершена сделка);
- price_tp - цена Take Profit (заполняется, если указана в ордере, на основе которого совершена сделка);
- volume - объем сделки в лотах.

© 2000-2025, MetaQuotes Ltd.

- position - тикет позиции, открытой, измененной или закрытой в результате исполнения сделки.
- position_by - тикет встречной позиции. Заполняется только для сделок на закрытие позиции встречной (out by).

TRADE_TRANSACTION_POSITION

Для торговых транзакций, касающихся изменений позиций, не связанных с исполнением сделок (TRADE_TRANSACTION_POSITION), в структуре MqlTradeTransaction заполняются следующие поля:

- symbol - имя финансового инструмента позиции;
- type - тип торговой транзакции;
- deal_type - тип позиции (DEAL_TYPE_BUY или DEAL_TYPE_SELL);
- price - средневзвешенная цена открытия позиции;
- price_sl - цена Stop Loss;
- price_tp - цена Take Profit;
- volume - объем позиции в лотах, если он был изменен.
- position - тикет позиции.

> Изменение позиции (добавление, изменение или ликвидация) в результате совершения сделки не влечет за собой появление транзакции TRADE_TRANSACTION_POSITION.

TRADE_TRANSACTION_REQUEST

Для торговых транзакций, описывающих факт, что торговый запрос обработан сервером, и результат его обработки получен (TRADE_TRANSACTION_REQUEST), в структуре MqlTradeTransaction заполняется только одно поле:

- type - тип торговой транзакции;

> Для транзакций данного типа необходимо анализировать только одно поле - type (тип торговой транзакции). Для получения дополнительной информации необходимо анализировать второй и третий параметры функции OnTradeTransaction (request и result).

Пример:

```
input int MagicNumber=1234567;

//--- подключим торговый класс CTrade и объявим переменную этого типа
#include <Trade\Trade.mqh>
CTrade trade;
//--- флаги для установки и удаления отложенного ордера
bool pending_done=false;
bool pending_deleted=false;
//--- здесь будем хранить тикет отложенного ордера
ulong order_ticket;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- установим MagicNumber, которым будут помечаться все наши ордера
```

© 2000-2025, MetaQuotes Ltd.

```
   trade.SetExpertMagicNumber(MagicNumber);
//--- торговые запросы будем отправлять в асинхронном режиме с помощью функции OrderSe
   trade.SetAsyncMode(true);
//--- инициализируем переменную нулем
   order_ticket=0;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- установка отложенного ордера
   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);
      //--- если функция BuyStop() отработала успешно
      if(res)
        {
         pending_done=true;
         //--- получим результат отправки запроса из ctrade
         MqlTradeResult trade_result;
         trade.Result(trade_result);
         //--- получим request_id для отправленного запроса
         uint request_id=trade_result.request_id;
         Print("Отправлен запрос на установку отложенного ордера. Идентификатор запрос
         //--- запомним тикет ордера (при использовании асинхронного режима отправки н
         order_ticket=trade_result.order;
         //--- всё сделано, поэтому досрочно выходим из обработчика OnTick()
         return;
        }
     }
//--- удаление отложенного ордера
   if(!pending_deleted)
      //--- дополнительная проверка
      if(pending_done && (order_ticket!=0))
        {
         //--- попытаемся удалить отложенный ордер
         bool res=trade.OrderDelete(order_ticket);
         Print("OrderDelete=",res);
         //--- при успешной отправке запроса на удаление
         if(res)
           {
            pending_deleted=true;
            //--- получим результат выполнения запроса
            MqlTradeResult trade_result;
```

© 2000-2025, MetaQuotes Ltd.

```
            trade.Result(trade_result);
            //--- вытащим из результата идентификатор запроса
            uint request_id=trade_result.request_id;
            //--- выведем в Журнал
            Print("Отправлен запрос на удаление отложенного ордера #",order_ticket,
                  ". Идентификатор запроса Request_ID=",request_id,
                  "\r\n");
            //--- запишем из результата запроса тикет ордера
            order_ticket=trade_result.order;
           }
         }
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- получим тип транзакции в виде значения перечисления
   ENUM_TRADE_TRANSACTION_TYPE type=(ENUM_TRADE_TRANSACTION_TYPE)trans.type;
//--- если транзакция является результатом обработки запроса, выведем только её назва
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      Print(EnumToString(type));
      //--- выведем строковое описание обработанного запроса
      Print("------------RequestDescription\r\n",RequestDescription(request));
      //--- выведем описание результата запроса
      Print("------------ResultDescription\r\n",TradeResultDescription(result));
      //--- запомним тикет ордера для его удаления на следующей обработке в OnTick()
      if(result.order!=0)
        {
         //--- удалим этот ордер по его тикету при следующем вызове OnTick()
         order_ticket=result.order;
         Print(" Тикет отложенного ордера ",order_ticket,"\r\n");
        }
     }
   else // для транзакций другого типа выведем полное описание
//--- выведем описание полученной транзакции в Журнал
      Print("------------TransactionDescription\r\n",TransactionDescription(trans));

//---
  }
//+------------------------------------------------------------------+
//| Возвращает текстовое описание транзакции                         |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   string desc=EnumToString(trans.type)+"\r\n";
   desc+="Symbol: "+trans.symbol+"\r\n";
   desc+="Deal ticket: "+(string)trans.deal+"\r\n";
   desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
   desc+="Order ticket: "+(string)trans.order+"\r\n";
   desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
   desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
   desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
   desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
   desc+="Price trigger: "+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";
   desc+="Position: "+(string)trans.position+"\r\n";
   desc+="Position by: "+(string)trans.position_by+"\r\n";
//--- вернем полученную строку
   return desc;
  }
//+------------------------------------------------------------------+
//| Возвращает текстовое описание торгового запроса                  |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request)
  {
//---
   string desc=EnumToString(request.action)+"\r\n";
   desc+="Symbol: "+request.symbol+"\r\n";
   desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
   desc+="Order ticket: "+(string)request.order+"\r\n";
   desc+="Order type: "+EnumToString(request.type)+"\r\n";
   desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
   desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
   desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
   desc+="Deviation points: "+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+="Comment: "+request.comment+"\r\n";
//--- вернем полученную строку
   return desc;
  }
//+------------------------------------------------------------------+
//| Возвращает текстовое описание результата обработки запроса       |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   string desc="Retcode "+(string)result.retcode+"\r\n";
   desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
   desc+="Order ticket: "+(string)result.order+"\r\n";
   desc+="Deal ticket: "+(string)result.deal+"\r\n";
   desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
   desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
   desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
   desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
   desc+="Comment: "+result.comment+"\r\n";
//--- вернем полученную строку
   return desc;
  }
```

Смотри также

Типы торговых транзакций, OnTradeTransaction()

© 2000-2025, MetaQuotes Ltd.

Структура для получения текущих цен (MqlTick)

Структура для хранения последних цен по символу. Предназначена для быстрого получения наиболее востребованной информации о текущих ценах.

```
struct MqlTick
  {
   datetime     time;          // Время последнего обновления цен
   double       bid;           // Текущая цена Bid
   double       ask;           // Текущая цена Ask
   double       last;          // Текущая цена последней сделки (Last)
   ulong        volume;        // Объем для текущей цены Last
   long         time_msc;      // Время последнего обновления цен в миллисекундах
   uint         flags;         // Флаги тиков
   double       volume_real;   // Объем для текущей цены Last с повышенной точностью
  };
```

Переменная типа MqlTick позволяет за один вызов функции SymbolInfoTick() получить значения Ask, Bid, Last и Volume.

У каждого тика всегда заполняются все параметры, независимо от того, изменились ли данные по сравнению с предыдущим тиком. Это позволяет всегда иметь актуальное состояние цен на любой момент времени без поиска предыдущих значений по тиковой истории. Например, с тиком могла измениться только цена бид, но в структуре помимо новой цены будут указаны и остальные параметры: предыдущая цена аск, объем и т.д.

Чтобы узнать, какие именно данные изменились с текущим тиком, анализируйте его флаги:

- TICK_FLAG_BID - тик изменил цену бид
- TICK_FLAG_ASK - тик изменил цену аск
- TICK_FLAG_LAST - тик изменил цену последней сделки
- TICK_FLAG_VOLUME - тик изменил объем
- TICK_FLAG_BUY - тик возник в результате сделки на покупку
- TICK_FLAG_SELL - тик возник в результате сделки на продажу

Пример:

```
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() failed, error = ",GetLastError());
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

Структуры и классы, CopyTicks(), SymbolInfoTick()

© 2000-2025, MetaQuotes Ltd.

Структуры экономического календаря

В этом разделе описываются структуры для работы с Экономическим календарем, который доступен прямо в платформе MetaTrader. Экономический календарь является готовой энциклопедией с описанием макроэкономических индикаторов, даты их выхода и степени важности. Актуальные значения макроэкономических показателей поступают в платформу MetaTrader моментально прямо в момент публикации и отображаются на графике в виде меток - это позволяет визаульно отслеживать нужные показатели в разрезе стран, валют и важности.

Функции Экономического календаря позволяют проводить автоматический анализ поступающих событий по собственным критериям важности и в разрезе нужных стран/валютных пар.

Описания стран задаются структурой MqlCalendarCountry. Используется в функциях CalendarCountryById() и CalendarCountries()

```
struct MqlCalendarCountry
  {
   ulong                                id;                  // идентификатор страны
   string                               name;                // текстовое имя страны
   string                               code;                // кодовое имя страны IS
   string                               currency;            // код валюты страны
   string                               currency_symbol;     // символ/знак валюты ст
   string                               url_name;            // имя страны, использу
  };
```

Описания событий задаются структурой MqlCalendarEvent. Используется в функциях CalendarEventById(), CalendarEventByCountry() и CalendarEventByCurrency()

```
struct MqlCalendarEvent
  {
   ulong                                id;                  // идентификатор события
   ENUM_CALENDAR_EVENT_TYPE             type;                // тип события из перечи
   ENUM_CALENDAR_EVENT_SECTOR           sector;              // сектор, к которому от
   ENUM_CALENDAR_EVENT_FREQUENCY        frequency;           // частота (периодичност
   ENUM_CALENDAR_EVENT_TIMEMODE         time_mode;           // режим времени события
   ulong                                country_id;          // идентификатор страны
   ENUM_CALENDAR_EVENT_UNIT             unit;                // единица измерения зна
   ENUM_CALENDAR_EVENT_IMPORTANCE       importance;          // важность события
   ENUM_CALENDAR_EVENT_MULTIPLIER       multiplier;          // множитель значения эн
   uint                                 digits;              // количество знаков пос
   string                               source_url;          // URL источника, где пу
   string                               event_code;          // код события
   string                               name;                // текстовое имя события
  };
```

© 2000-2025, MetaQuotes Ltd.

Значения событий задаются структурой MqlCalendarValue. Используется в функциях CalendarValueById(), CalendarValueHistoryByEvent(), CalendarValueHistory(), CalendarValueLastByEvent() и CalendarValueLast()

```
struct MqlCalendarValue
  {
   ulong                             id;                // ID значения
   ulong                             event_id;          // ID события
   datetime                          time;              // время и дата события
   datetime                          period;            // отчетный период собы
   int                               revision;          // ревизия публикуемого
   long                              actual_value;      // актуальное значение
   long                              prev_value;        // предыдущее значение
   long                              revised_prev_value; // пересмотренное преды
   long                              forecast_value;    // прогнозное значение
   ENUM_CALENDAR_EVENT_IMPACT        impact_type;       // потенциальное влияние
   //--- функции для проверки значений
   bool                           HasActualValue(void) const;   // возвращет true, если
   bool                           HasPreviousValue(void) const; // возвращет true, если
   bool                           HasRevisedValue(void) const;  // возвращет true, если
   bool                           HasForecastValue(void) const; // возвращет true, если
   //--- функции для получение значений
   double                         GetActualValue(void) const;   // возвращает actual_val
   double                         GetPreviousValue(void) const; // возвращает prev_value
   double                         GetRevisedValue(void) const;  // возвращает revised_p
   double                         GetForecastValue(void) const; // возвращает forecast_
  };
```

Структура MqlCalendarValue предоставляет методы для проверки и получения значений из полей *actual_value*, *forecast_value*, *prev_value* и *revised_prev_value*. Если значение поля не задано, то поле хранит значение LONG_MIN (-9223372036854775808).

При этом необходимо иметь в виду, что значения в этих полях хранятся увеличенными в миллион раз. Это означает, что при получении значений в MqlCalendarValue функциями CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent и CalendarValueLast, значения этих полей нужно проверять на равенство LONG_MIN, и если значение в поле задано, то для получения значения необходимо разделить значение поля на 1 000 000 (миллион). Другой способ получения значений - делать проверку и получать значения функциями самой структуры MqlCalendarValue.

Пример обработки событий календаря:

```
//--- создадим свою структуру под хранение событий календаря с вещественными значениям
struct AdjustedCalendarValue
  {
   ulong                             id;                // ID значения
   ulong                             event_id;          // ID события
   datetime                          time;              // время и дата события
   datetime                          period;            // отчетный период собы
   int                               revision;          // ревизия публикуемого
```

© 2000-2025, MetaQuotes Ltd.

```
   double                              actual_value;          // актуальное значение п
   double                              prev_value;            // предыдущее значение п
   double                              revised_prev_value;    // пересмотренное предыд
   double                              forecast_value;        // прогнозное значение п
   ENUM_CALENDAR_EVENT_IMPACT          impact_type;           // потенциальное влияние
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
//--- код страны для Европейского союза по стандарту ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- получим все значения событий по Европейскому союзу
   MqlCalendarValue values[];
//--- зададим границы диапазона, из которого берем события
   datetime date_from=D'01.01.2021';  // берем все события с 2021 года
   datetime date_to=0;                // 0 означает все известные события, даже те, чт
//--- запросим историю события по Европейскому союзу с 2021 года
   if(!CalendarValueHistory(values, date_from, date_to, EU_code))
     {
      PrintFormat("Ошибка! Не удалось получить события по стране country_code=%s", EU_
      PrintFormat("Код ошибки: %d", GetLastError());
      return;
     }
   else
      PrintFormat("Получены значения событий по country_code=%s: %d",
                  EU_code, ArraySize(values));
//--- уменьшим размер массива для вывода в Журнал
   if(ArraySize(values)>5)
      ArrayResize(values, 5);
//--- выведем значения событий в Журнал как есть, без проверок и приведения к актуальн
   Print("Выводим значения календаря как есть");
   ArrayPrint(values);

//--- проверим значения полей и приведем к актуальным значениям
//--- первый вариант проверок и получения значений
   AdjustedCalendarValue values_adjusted_1[];
   int total=ArraySize(values);
   ArrayResize(values_adjusted_1, total);
//--- копируем значения с проверками и корректировками
   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;
```

© 2000-2025, MetaQuotes Ltd.

```
      values_adjusted_1[i].impact_type=values[i].impact_type;
      //--- делаем проверки значений и делим на 1 000 000
      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("Первый способ проверки и получения значений календаря");
   ArrayPrint(values_adjusted_1);

//--- второй вариант проверок и получения значений
   AdjustedCalendarValue values_adjusted_2[];
   ArrayResize(values_adjusted_2, total);
//--- копируем значения с проверками и корректировками
   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;
      //--- делаем проверки и получаем значения
      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())
```

© 2000-2025, MetaQuotes Ltd.

```
         values_adjusted_2[i].revised_prev_value=values[i].GetRevisedValue();
      else
         values_adjusted_2[i].revised_prev_value=double("nan");

      if(values[i].HasForecastValue())
         values_adjusted_2[i].forecast_value=values[i].GetForecastValue();
      else
         values_adjusted_2[i].forecast_value=double("nan");
     }
   Print("Второй способ проверки и получения значений календаря");
   ArrayPrint(values_adjusted_2);

//--- третий вариант получения значений - без проверок
   AdjustedCalendarValue values_adjusted_3[];
   ArrayResize(values_adjusted_3, total);
//--- копируем значения с проверками и корректировками
   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;
      //--- получаем значения без проверок
      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("Третий способ получения значений календаря - без проверок");
   ArrayPrint(values_adjusted_3);
  }
/*
   Получены значения событий по country_code=EU: 1051
   Выводим значения календаря как есть
         [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
   Первый способ проверки и получения значений календаря
         [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
```

© 2000-2025, MetaQuotes Ltd.

```
   Второй способ проверки и получения значений календаря
         [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
   Третий способ получения значений календаря - без проверок
         [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
*/
```

Частота (периодичность) события указывается в структуре MqlCalendarEvent. Возможные значения указаны в перечислении **ENUM_CALENDAR_EVENT_FREQUENCY**

Идентификатор	Описание
CALENDAR_FREQUENCY_NONE	Частота публикации не задана
CALENDAR_FREQUENCY_WEEK	Публикация один раз в неделю
CALENDAR_FREQUENCY_MONTH	Публикация один раз в месяц
CALENDAR_FREQUENCY_QUARTER	Публикация один раз в квартал
CALENDAR_FREQUENCY_YEAR	Публикация один раз в год
CALENDAR_FREQUENCY_DAY	Публикация один раз в день

Тип события указывается в структуре MqlCalendarEvent. Возможные значения указаны в перечислении **ENUM_CALENDAR_EVENT_TYPE**

Идентификатор	Описание
CALENDAR_TYPE_EVENT	Событие (митинг, речь и так далее)
CALENDAR_TYPE_INDICATOR	Индикатор
CALENDAR_TYPE_HOLIDAY	Праздник

Сектор экономики, к которому относится событие, указывается в структуре MqlCalendarEvent. Возможные значения указаны в перечислении **ENUM_CALENDAR_EVENT_SECTOR**

Идентификатор	Описание
CALENDAR_SECTOR_NONE	Сектор не задан

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
CALENDAR_SECTOR_MARKET	Рынок, биржа
CALENDAR_SECTOR_GDP	Валовый внутренний продукт (GDP)
CALENDAR_SECTOR_JOBS	Рынок труда
CALENDAR_SECTOR_PRICES	Цены
CALENDAR_SECTOR_MONEY	Деньги
CALENDAR_SECTOR_TRADE	Торговля
CALENDAR_SECTOR_GOVERNMENT	Правительство
CALENDAR_SECTOR_BUSINESS	Бизнес
CALENDAR_SECTOR_CONSUMER	Потребление
CALENDAR_SECTOR_HOUSING	Жилье
CALENDAR_SECTOR_TAXES	Налоги
CALENDAR_SECTOR_HOLIDAYS	Праздники

Важность события указывается в структуре MqlCalendarEvent. Возможные значения указаны в перечислении **ENUM_CALENDAR_EVENT_IMPORTANCE**

Идентификатор	Описание
CALENDAR_IMPORTANCE_NONE	Степень важности не задана
CALENDAR_IMPORTANCE_LOW	Низкая важность
CALENDAR_IMPORTANCE_MODERATE	Средняя важность
CALENDAR_IMPORTANCE_HIGH	Высокая важность

Тип единицы измерения, в которых даются значения события, указывается в структуре MqlCalendarEvent. Возможные значения указаны в перечислении **ENUM_CALENDAR_EVENT_UNIT**

Идентификатор	Описание
CALENDAR_UNIT_NONE	Единица измерения не задана
CALENDAR_UNIT_PERCENT	Проценты
CALENDAR_UNIT_CURRENCY	Национальная валюта
CALENDAR_UNIT_HOUR	Количество часов
CALENDAR_UNIT_JOB	Количество рабочих мест
CALENDAR_UNIT_RIG	Буровые установки

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
CALENDAR_UNIT_USD	Доллары США
CALENDAR_UNIT_PEOPLE	Число людей
CALENDAR_UNIT_MORTGAGE	Количество ипотечных кредитов
CALENDAR_UNIT_VOTE	Число голосов
CALENDAR_UNIT_BARREL	Количество баррелях
CALENDAR_UNIT_CUBICFEET	Объем в кубических футах
CALENDAR_UNIT_POSITION	Чистый объем спекулятивных позиций в контрактах
CALENDAR_UNIT_BUILDING	Количество строений

В некоторых случаях значения экономического показателя требуют указания множителя, который указывается в структуре MqlCalendarEvent. Возможные значения множителей указаны в перечислении **ENUM_CALENDAR_EVENT_MULTIPLIER**

Идентификатор	Описание
CALENDAR_MULTIPLIER_NONE	Множитель не задан
CALENDAR_MULTIPLIER_THOUSANDS	Тысячи
CALENDAR_MULTIPLIER_MILLIONS	Миллионы
CALENDAR_MULTIPLIER_BILLIONS	Миллиарды
CALENDAR_MULTIPLIER_TRILLIONS	Триллионы

Потенциальное влияние события на курс национальной валюты указывается в структуре MqlCalendarValue. Возможные значения указаны в перечислении **ENUM_CALENDAR_EVENT_IMPACT**

Идентификатор	Описание
CALENDAR_IMPACT_NA	Влияния не указано
CALENDAR_IMPACT_POSITIVE	Положительное влияние
CALENDAR_IMPACT_NEGATIVE	Отрицательное влияние

Время наступления события указывается в структуре MqlCalendarEvent. Возможные значения указаны в перечислении **ENUM_CALENDAR_EVENT_TIMEMODE**

Идентификатор	Описание
CALENDAR_TIMEMODE_DATETIME	Источник публикует точное время

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
	наступления события
CALENDAR_TIMEMODE_DATE	Событие занимает весь день
CALENDAR_TIMEMODE_NOTIME	Источник не публикует время события
CALENDAR_TIMEMODE_TENTATIVE	Источник не публикует предварительно точное время события, только его день. Время уточняется по факту наступления события

Смотри также

Экономический календарь

© 2000-2025, MetaQuotes Ltd.

Коды ошибок и предупреждений

Раздел содержит следующие описания:

- Коды возврата торгового сервера - анализ результатов отправки торгового запроса, отправленного функцией OrderSend();
- Предупреждения компилятора - коды предупредительных сообщений, выводимые при компиляции (не являются ошибками);
- Ошибки компиляции - коды сообщений об ошибке при неудачной попытке компиляции;
- Ошибки времени выполнения - коды ошибок при выполнении mql5-программы, которые можно получить при помощи функции GetLastError().

© 2000-2025, MetaQuotes Ltd.

Коды возврата торгового сервера

Все приказы на совершение торговых операций отправляются в виде структуры торгового запроса MqlTradeRequest с помощью функции OrderSend(). Результат выполнения этой функции помещается в структуру MqlTradeResult, поле *retcode* которой содержит код возврата торгового сервера.

Код	Идентификатор	Описание
10004	TRADE_RETCODE_REQUOTE	Реквота
10006	TRADE_RETCODE_REJECT	Запрос отклонен
10007	TRADE_RETCODE_CANCEL	Запрос отменен трейдером
10008	TRADE_RETCODE_PLACED	Ордер размещен
10009	TRADE_RETCODE_DONE	Заявка выполнена
10010	TRADE_RETCODE_DONE_PARTIAL	Заявка выполнена частично
10011	TRADE_RETCODE_ERROR	Ошибка обработки запроса
10012	TRADE_RETCODE_TIMEOUT	Запрос отменен по истечению времени
10013	TRADE_RETCODE_INVALID	Неправильный запрос
10014	TRADE_RETCODE_INVALID_VOLUME	Неправильный объем в запросе
10015	TRADE_RETCODE_INVALID_PRICE	Неправильная цена в запросе
10016	TRADE_RETCODE_INVALID_STOPS	Неправильные стопы в запросе
10017	TRADE_RETCODE_TRADE_DISABLED	Торговля запрещена
10018	TRADE_RETCODE_MARKET_CLOSED	Рынок закрыт
10019	TRADE_RETCODE_NO_MONEY	Нет достаточных денежных средств для выполнения запроса
10020	TRADE_RETCODE_PRICE_CHANGED	Цены изменились
10021	TRADE_RETCODE_PRICE_OFF	Отсутствуют котировки для обработки запроса
10022	TRADE_RETCODE_INVALID_EXPIRATION	Неверная дата истечения ордера в запросе
10023	TRADE_RETCODE_ORDER_CHANGED	Состояние ордера изменилось
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	Слишком частые запросы
10025	TRADE_RETCODE_NO_CHANGES	В запросе нет изменений
10026	TRADE_RETCODE_SERVER_DISABLES_AT	Автотрейдинг запрещен сервером
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	Автотрейдинг запрещен клиентским терминалом
10028	TRADE_RETCODE_LOCKED	Запрос заблокирован для обработки

© 2000-2025, MetaQuotes Ltd.

Код	Идентификатор	Описание
10029	TRADE_RETCODE_FROZEN	Ордер или позиция заморожены
10030	TRADE_RETCODE_INVALID_FILL	Указан неподдерживаемый тип исполнения ордера по остатку
10031	TRADE_RETCODE_CONNECTION	Нет соединения с торговым сервером
10032	TRADE_RETCODE_ONLY_REAL	Операция разрешена только для реальных счетов
10033	TRADE_RETCODE_LIMIT_ORDERS	Достигнут лимит на количество отложенных ордеров
10034	TRADE_RETCODE_LIMIT_VOLUME	Достигнут лимит на объем ордеров и позиций для данного символа
10035	TRADE_RETCODE_INVALID_ORDER	Неверный или запрещённый тип ордера
10036	TRADE_RETCODE_POSITION_CLOSED	Позиция с указанным POSITION_IDENTIFIER уже закрыта
10038	TRADE_RETCODE_INVALID_CLOSE_VOLUME	Закрываемый объем превышает текущий объем позиции
10039	TRADE_RETCODE_CLOSE_ORDER_EXIST	Для указанной позиции уже есть ордер на закрытие. Может возникнуть при работе в системе хеджинга: • при попытке закрытия позиции встречной, если уже есть ордера на закрытие этой позиции • при попытке полного или частичного закрытия, если суммарный объем уже имеющихся ордеров на закрытие и вновь выставляемого ордера превышает текущий объем позиции
10040	TRADE_RETCODE_LIMIT_POSITIONS	Количество открытых позиций, которое можно одновременно иметь на счете, может быть ограничено настройками сервера. При достижении лимита в ответ на выставление ордера сервер вернет ошибку TRADE_RETCODE_LIMIT_POSITIONS. Ограничение работает по-разному в зависимости от типа учета позиций на счете: • Неттинговая система — учитывается количество открытых позиции. При достижении лимита платформа не позволит выставлять новые ордера, в результате исполнения которых может увеличиться количество открытых позиций. Фактически, платформа

© 2000-2025, MetaQuotes Ltd.

Код	Идентификатор	Описание
		позволит выставлять ордера только по тем символам, по которым уже есть открытые позиции. В неттинговой системе при проверке лимита не учитываются текущие отложенные ордера, поскольку их исполнение может привести к изменению текущих позиций, а не увеличению их количества. • Хеджинговая система — помимо открытых позиций, учитываются выставленные отложенные ордера, поскольку их срабатывание всегда приводит к открытию новой позиции. При достижении лимита платформа не позволит выставлять рыночные ордера на открытие позиций, а также отложенные ордера.
10041	TRADE_RETCODE_REJECT_CANCEL	Запрос на активацию отложенного ордера отклонен, а сам ордер отменен
10042	TRADE_RETCODE_LONG_ONLY	Запрос отклонен, так как на символе установлено правило "**Разрешены только длинные позиции**" (POSITION_TYPE_BUY)
10043	TRADE_RETCODE_SHORT_ONLY	Запрос отклонен, так как на символе установлено правило "**Разрешены только короткие позиции**" (POSITION_TYPE_SELL)
10044	TRADE_RETCODE_CLOSE_ONLY	Запрос отклонен, так как на символе установлено правило "**Разрешено только закрывать существующие позиции**"
10045	TRADE_RETCODE_FIFO_CLOSE	Запрос отклонен, так как для торгового счета установлено правило "**Разрешено закрывать существующие позиции только по правилу FIFO**" (ACCOUNT_FIFO_CLOSE=true)
10046	TRADE_RETCODE_HEDGE_PROHIBITED	Запрос отклонен, так как для торгового счета установлено правило "**Запрещено открывать встречные позиции по одному символу**". Например, если на счете имеется позиция Buy, то пользователь не может открыть позицию Sell или выставить отложенный ордер на продажу. Правило может применяться только на счетах с хеджинговой

© 2000-2025, MetaQuotes Ltd.

Код	Идентификатор	Описание
		системой учета (ACCOUNT_MARGIN_MODE=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING).

© 2000-2025, MetaQuotes Ltd.

Предупреждения компилятора

Предупреждения компилятора носят информационный характер и не являются сообщениями об ошибках.

Номер	Описание
21	Неполная запись даты в строке datetime
22	Ошибочные числа в строке datetime для даты, требования: год 1970<=X<=3000 месяц 0<X<=12 день 0<X<= 31/30/28(29)....
23	Ошибочные числа в строке datetime для времени, требования: час 0<=X<24 минута 0<=X<60
24	Некорректный цвет в формате RGB: одна из компонент RGB меньше 0 или больше 255
25	Неизвестный символ эскейп последовательности. Известные: \n \r \t \\ \" \' \X \x
26	Слишком большой объем локальных переменных (>512кб) функции, уменьшите их количество
29	Перечисление уже определено (дублирование) - члены будут добавлены к первому определению
30	Переопределение макроса
31	Переменная объявлена, но нигде не используется
32	Конструктор должен иметь тип void
33	Деструктор должен иметь тип void
34	Константа не вмещается в диапазон целых (X>_UI64_MAX || X<_I64_MIN) и будет преобразована в тип double
35	Слишком длинный HEX больше 16 значащих символов (обрезаются старшие полубайты)
36	Нет ни одного полубайта в HEX строке "0x"
37	Нет ни одной функции - нечего будет выполнять
38	Используется неинициализированная переменная
41	Функция не имеет тела, но и не вызывается
43	Возможны потери данных при преобразовании типа. Пример: int x=(double)z;
44	Потеря точности(данных) при преобразовании константы. Пример: int x=M_PI
45	Несовпадение знаков операндов в операциях сравнения. Пример: (char) c1>(uchar)c2

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
46	Проблемы с импортом функций - требуется объявление #import либо импорт функций уже закрыт
47	Описание слишком большое - лишние символы не будут включены в исполняемый файл
48	Количество индикаторных буферов объявлено меньше, чем требуется
49	Не указан цвет для отрисовки графической серии в индикаторе
50	Нет ни одной графической серии для отображения индикатора
51	Не обнаружена функция-обработчик 'OnStart" в скрипте
52	Функция-обработчик 'OnStart" определена с неверными параметрами
53	Функция 'OnStart' может быть определена только в скрипте
54	Функция 'OnInit' определена с неверными параметрами
55	Функция 'OnInit' не используется в скриптах
56	Функция 'OnDeinit' определена с неверными параметрами
57	Функция 'OnDeinit' не используется в скриптах
58	Определены две функции 'OnCalculate'. Будет использована OnCalculate() на одном ценовом массиве
59	Обнаружено переполнение при вычислении сложной целочисленной константы
60	Возможно, переменная неинициализирована.
61	Данное объявление делает недоступным обращение к локальной переменной, объявленной на указанной строке
62	Данное объявление делает недоступным обращение к глобальной переменной, объявленной на указанной строке
63	Не может быть использовано для статических массивов
64	Данное объявление делает недоступным обращение к предопределенной переменной
65	Значение выражения всегда true/false
66	Использование переменной или выражения типа bool в математических операциях является небезопасным
67	Результат применения оператора унарного минуса к беззнаковому типу ulong неопределен
68	Версия, указанная в свойстве #property version, недопустима для размещения в разделе Маркет, правильный формат #property version "XXX.YYY"
69	Отсутствует выражение для выполнения по условию

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
70	Неверный возвращаемый тип функции или некорректные параметры при объявлении функции-обработчика события
71	Требуется явное приведение структур к одному типу
72	Данное объявление делает недоступным прямое обращение к члену класса, объявленному на указанной строке. Доступ будет возможен только с помощью операции разрешения контекста ::
73	Константа в двоичной записи слишком велика, старшие разряды будут отброшены
74	Параметр в методе наследуемого класса отличается модификатором const, дочерняя функция перегрузила функцию родителя
75	Отрицательное или слишком большое значения смещения в битовой операции сдвига, результат выполнения неопределён
76	Функция должна вернуть значение
77	Функция типа void не должна возвращать значение
78	Не все варианты выполнения возвращают значение
79	Выражения на глобальном уровне не разрешены
80	Возможна ошибка в последовательности выполнения операций, используйте скобки для явного указания порядка
81	Найдено два вида вызова OnCalCulate(). Вызываться будет вариант с использованием таймсерий OHLC
82	Структура не содержит членов, размер будет приравнен 1 байту
83	Нет обработки результата выполнения функции
84	Индикатор, включаемый как ресурс, скомпилирован в режиме отладки. Это снижает его производительность. Для повышения скорости работы его нужно перекомпилировать
85	Слишком большой код символа в строке, должен быть в диапазоне от 0 до 65535
86	Нераспознанный служебный символ в строке
87	Не указано свойство индикатора, задающее вывод в главное окно или в отдельное подокно. Будет применено свойство #property indicator_chart_window

© 2000-2025, MetaQuotes Ltd.

Ошибки компиляции

MetaEdtior 5, редактор mql5-программ, выдает сообщения об ошибках программы, обнаруженных встроенным компилятором на стадии компиляции. Список этих ошибок приведен ниже в таблице. Для компиляции исходного кода в исполняемый нажмите **F7**. Программы с ошибками не могут быть скомпилированы, пока ошибки, указанные компилятором, не будут устранены.

Номер	Описание
100	Ошибка чтения файла
101	Ошибка открытия *.EX5 файла для записи
103	Недостаточно свободной памяти для завершения компиляции
104	Нераспознанная компилятором пустая синтаксическая единица
105	Некорректное имя файла в #include
106	Ошибка доступа к файлу в #include (возможно файл не существует)
108	Неподходящее имя для #define
109	Неизвестная команда препроцессора (допустимы #include,#define,#property,#import)
110	Неизвестный для компилятора символ
111	Функция не реализована (описание есть, тела нет)
112	Пропущена двойная кавычка (")
113	Пропущена открывающая угловая скобка (<) или двойная кавычка (")
114	Пропущена одинарная кавычка (')
115	Пропущена закрывающая угловая скобка ">"
116	Не указан тип в объявлении
117	Нет оператора возврата return или имеется не во всех ветках выполнения
118	Ожидалась открывающая скобка параметров вызова
119	Ошибка записи EX5
120	Некорректный доступ к элементу массива
121	Функция не имеет тип void и оператор return должен вернуть значение
122	Некорректное объявление деструктора
123	Отсутствует двоеточие ":"
124	Переменная уже объявлена
125	Переменная с таким идентификатором уже объявлена
126	Имя переменной слишком длинное (>250 символов)
127	Структура с таким идентификатором уже определена

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
128	Структура не определена
129	Член структуры с таким именем уже определен
130	Нет такого члена структуры
131	Нарушена парность квадратных скобок
132	Ожидается открывающая круглая скобка "("
133	Несбалансированные фигурные скобки (отсутствует "}")
134	Сложно для компиляции (слишком большое ветвление, внутренний стек уровней переполнен)
135	Ошибка открытия файла на чтение
136	Недостаточно памяти для загрузки исходного файла в память
137	Ожидается переменная
138	Ссылка не может быть инициализирована
140	Ожидалось присваивание (возникает при объявлении)
141	Ожидается открывающая фигурная скобка "{"
142	Параметр может быть только динамическим массивом
143	Использование типа "void" недопустимо
144	Нет пары для ")" или "]", т.е. отсутствует "(" или "["
145	Нет пары для "(" или "[", т.е. отсутствует ")" или "]"
146	Некорректная размерность массива
147	Слишком много параметров (>64)
149	Этот токен тут не ожидается
150	Недопустимое использование операции (неправильные операнды)
151	Выражение типа void недопустимо
152	Ожидается оператор
153	Неправильное использование break
154	Ожидается точка с запятой ";"
155	Ожидается запятая ","
156	Тип должен быть определен как класс, а не как структура
157	Ожидалось выражение
158	В HEX встречается "не HEX символ" или слишком длинное число (количество цифр > 511)

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
159	Строка-константа имеет более 65534 символов
160	Определение функции здесь недопустимо
161	Неожиданный конец программы
162	Форвардная декларация для структур запрещена
163	Функция с таким именем уже определена и имеет иной тип возвращаемого значения
164	Функция с таким именем уже определена и имеет иной набор параметров
165	Функция с таким именем уже определена и реализована
166	Перегрузка функции для данного вызова не найдена
167	Функция с возвращаемым значением типа void не может возвращать значение
168	Функция не определена
170	Ожидается значение
171	В выражении case допустимы только целочисленные константы
172	Значение для case в этом switch уже использовано
173	Ожидается целочисленное значение
174	В выражении #import ожидается имя файла
175	Выражения на глобальном уровне не допустимы
176	Пропущена круглая скобка ")" перед ";"
177	Слева от знака равенства предполагается переменная
178	Результат выражения не используется
179	Объявление переменных в case недопустимо
180	Неявное преобразование из строки в число
181	Неявное преобразование числа в строку
182	Неоднозначный вызов перегруженной функции (подходят несколько перегрузок)
183	Недопустимый else без соответствующего if
184	Недопустимый case или default без соответствующего switch
185	Недопустимое использование эллипсиса
186	Инициализирующая последовательность имеет большее количество элементов чем инициализируемая переменная
187	Ожидается константа для case
188	Требуется константное выражение

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
189	Константная переменная не может быть изменена
190	Ожидается закрывающая скобка или запятая (объявление члена массива)
191	Идентификатор перечисления уже используется
192	Перечисление не может иметь модификаторов доступа (const, extern, static)
193	Член перечисления уже объявлен с другим значением
194	Существует переменная, определенная с таким же именем
195	Существует структура, определенная с таким же именем
196	Ожидается имя члена перечисления
197	Ожидается целочисленное выражение
198	Деление на ноль в константном выражении
199	Неверное количество параметров в функции
200	Параметром по ссылке должна быть переменная
201	Ожидается переменная такого же типа для передачи по ссылке
202	Константная переменная не может быть передана по неконстантной ссылке
203	Требуется целочисленная положительная константа
204	Ошибка доступа к защищенному члену класса
205	Импорт уже определен по другому пути
208	Исполняемый файл не создан
209	Для индикатора не найдена точка входа 'OnCalculate'
210	Оператор continue может быть использован только внутри цикла
211	Ошибка доступа к private(закрытому) члену класса
213	Метод структуры или класса не объявлен
214	Ошибка доступа к private(закрытому) методу класса
216	Копирование структур с объектами недопустимо
218	Выход индекса за границы массива
219	Недопустима инициализация массивов в объявлении структуры или класса
220	Конструктор класса не может иметь параметров
221	Деструктор класса не может иметь параметров
222	Метод класса или структуры с таким именем и параметрами уже объявлен
223	Ожидается операнд

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
224	Метод класса или структуры с таким именем есть, но с другими параметрами (объявление!=реализация)
225	Импортируемая функция не описана
226	Функция ZeroMemory() не применима для классов с защищенными членами или наследованием
227	Неоднозначный вызов перегруженной функции (точное совпадение параметров для нескольких перегрузок)
228	Ожидается имя переменной
229	Ссылку нельзя объявить в этом месте
230	Уже используется в качестве имени перечисления
232	Ожидается класс или структура
235	Нельзя вызывать delete для удаления массива
236	Ожидается оператор ' while '
237	В delete должен быть указатель
238	default для этого switch уже есть
239	Синтаксическая ошибка
240	Escape-последовательность может встретиться только в строках (начинается с '\')
241	Требуется массив - квадратная скобка '[' не относится к массиву либо в качестве параметра-массива подают не массив
242	Не может быть инициализировано посредством инициализирующей последовательности
243	Импорт не определен
244	Ошибка оптимизатора на синтаксическом дереве
245	Объявлено слишком много структур (упростите программу)
246	Преобразование параметра недопустимо
247	Некорректное использование оператора delete
248	Нельзя объявить указатель на ссылку
249	Нельзя объявить ссылку на ссылку
250	Нельзя объявить указатель на указатель
251	Недопустимо объявление структуры в списке параметров
252	Недопустимая операция приведения типов
253	Указатель можно объявить только для класса или структуры

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
256	Необъявленный идентификатор
257	Ошибка оптимизатора исполняемого кода
258	Ошибка генерации исполняемого кода
260	Недопустимое выражение для оператора switch
261	Переполнение пула строковых констант, упростите программу
262	Невозможно преобразовать к перечислению
263	Нельзя использовать virtual для данных (членов класса или структуры)
264	Нельзя вызвать защищенный метод класса
265	Переопределяемая виртуальная функция возвращает другой тип
266	Класс нельзя наследовать от структуры
267	Структуру нельзя наследовать от класса
268	Конструктор не может быть виртуальным (спецификатор virtual недопустим)
269	Структура не может иметь виртуальных методов
270	Функция должна иметь тело
271	Перегрузка системных функций (функций терминала) запрещена
272	Спецификатор const недопустим для функций, не являющихся членом класса или структуры
274	Нельзя менять члены класса в константном методе
276	Неподходящая инициализирующая последовательность
277	Пропущено значение по умолчанию для параметра (специфика объявления параметров по умолчанию)
278	Переопределение параметра по умолчанию (в объявлении и реализации разные значения)
279	Нельзя вызвать неконстантный метод для константного объекта
280	Для доступа к членам требуется объект (поставлена точка для не класса/структуры)
281	Имя уже объявленной структуры нельзя использовать при объявлении
284	Неразрешенное преобразование (при закрытом наследовании)
285	Структуры и массивы не могут быть использованы в качестве input-переменных
286	Спецификатор const недопустим для конструктора/деструктора
287	Неправильное строковое выражение для типа datetime
288	Неизвестное свойство (#property)

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
289	Некорректное значение для свойства
290	Некорректный индекс для свойства в #property
291	Пропущен параметр вызова - < func(x,) >
293	Объект должен быть передан по ссылке
294	Массив должен быть передан по ссылке
295	Функция была декларирована как экспортируемая
296	Функция не была декларирована как экспортируемая
297	Экспортировать импортируемую функцию нельзя
298	Импортируемая функция не может иметь такого параметра (нельзя передавать указатель, класс или структуру, содержащую динамический массив, указатель, класс и т.д.)
299	Должен быть класс
300	Секция #import не закрыта
302	Несоответствие типов
303	extern-переменная уже инициализирована
304	Не найдено ни одной экспортируемой функции или стандартной точки входа
305	Явный вызов конструктора запрещен
306	Метод был объявлен константным
307	Метод не был объявлен константным
308	Некорректный размер ресурсного файла
309	Некорректное имя ресурса
310	Ошибка открытия файла ресурса
311	Ошибка чтения файла ресурса
312	Неизвестный тип ресурса
313	Некорректный путь к файлу ресурса
314	Указанное имя ресурса уже используется
315	Ожидались параметры макроса
316	После имени макроса должен быть пробел
317	Ошибка в описании параметров макроса
318	Неверное число параметров при использовании макроса
319	Превышение максимального количества(16) параметров для макроса

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
320	Макрос слишком сложный, требуется упрощение
321	Параметром EnumToString() может быть только перечисление
322	Имя ресурса слишком длинное
323	Неподдерживаемый формат изображения (допустим только BMP-формат с глубиной цвета 24 или 32 бита)
324	Объявление массива внутри оператора запрещено
325	Функцию можно определить только на глобальном уровне
326	Данное объявление недопустимо для текущей области видимости (области определения)
327	Инициализация статичных переменных значениями локальных недопустима
328	Недопустимое объявление массива объектов, не имеющих конструктора по умолчанию
329	Список инициализации разрешен только для конструкторов
330	Отсутствует определение функции после списка инициализации
331	Список инициализации пуст
332	Инициализация массива в конструкторе запрещена
333	В списке инициализации запрещено инициализировать члены родительского класса
334	Ожидалось выражение целого типа
335	Требуемый объем памяти для массива превышает максимально допустимое значение
336	Требуемый объем памяти для структуры превышает максимально допустимое значение
337	Требуемый объем памяти для переменных, объявленных на глобальном уровне, превышает максимально допустимое значение
338	Требуемый объем памяти для локальных переменных превышает максимально допустимое значение
339	Конструктор не определен
340	Недопустимое имя для файла иконки
341	Не удалось открыть файла иконки по указанному пути
342	Файл иконки некорректен и не соответствует формату ICO
343	Повторная инициализация члена в конструкторе класса/структуры с помощью списка инициализации

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
344	Инициализация статических членов в списке инициализации конструктора не допускается
345	Инициализация нестатического члена класса/структуры на глобальном уровне запрещена
346	Имя метода класса/структуры совпадает с ранее объявленным именем члена
347	Имя члена класса/структуры совпадает с ранее объявленным именем метода
348	Виртуальная функция не может быть объявлена как static
349	Модификатор const недопустим для статической функции
350	Конструктор или деструктор не могут быть статическими
351	Нельзя обращаться к нестатическому члену/методу класса или структуры из статической функции
352	После ключевого слова operator ожидается перегружаемая операция (+,-,[],++,-- и т.д.)
353	Не все операции можно перегружать в MQL5
354	Определение не соответствует объявлению
355	Указано неверное количество параметров для оператора
356	Не обнаружено ни одной функции-обработчика события
357	Методы не могут быть экспортируемыми
358	Нельзя приводить указатель на константный объект к указателю на неконстантный объект
359	Шаблоны классов пока не поддерживаются
360	Перегрузка шаблонов функций пока не поддерживается
361	Невозможно применить шаблон функции
362	Неоднозначный параметр в шаблоне функции (подходят несколько типов параметра)
363	Невозможно определить к какому типу параметра приводить аргумент шаблона функции
364	Неверное количество параметров в шаблоне функции
365	Шаблон функции не может быть виртуальным
366	Шаблоны функций не могут быть экспортированы
367	Нельзя импортировать шаблоны функций
368	Структуры, содержащие объекты, недопустимы
369	Массивы строк и структуры, содержащие объекты, недопустимы

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
370	Статический член класса/структуры должен быть явно инициализирован
371	Ограничение компилятора: строка не может содержать более 65 535 символов
372	Несогласованные #ifdef/#endif
373	Результатом выполнения функции не может быть объект класса, так как отсутствует конструктор копирования
374	Нельзя использовать нестатические члены и/или методы при инициализации статической переменной
375	OnTesterInit() нельзя использовать без объявления обработчика OnTesterDeinit()
376	Имя локальной переменной совпадает с именем одного из параметров функции
377	Нельзя использовать макросы __FUNCSIG__ и __FUNCTION__ вне тела функции
378	Недопустимый возвращаемый тип. Например, такая ошибка будет выдана для функций, импортированных из DLL, которые возвращают структуру или указатель в качестве результата
379	Ошибка при использовании шаблона
380	Не используется
381	Недопустимый синтаксис при объявлении чисто виртуальной функции, разрешено "=NULL" или "=0"
382	Только виртуальные функции могут быть объявлены со спецификатором чисто виртуальной функции ("=NULL" или "=0")
383	Нельзя создать экземпляр абстрактного класса
384	Для динамического приведения с помощью оператора dynamic_cast типом назначения должен быть указатель на пользовательский тип
385	Ожидается тип "указатель на функцию"
386	Указатели на методы не поддерживаются
387	Ошибка - невозможно определить тип указателя на функцию
388	Приведение типа недоступно из-за закрытого наследования
389	Переменная с модификатором const должна быть проинициализирована при объявлении
393	В интерфейсе могут быть объявлены только методы с публичным доступом
394	Недопустимое вложение интерфейса в другой интерфейс
395	Интерфейс может наследоваться только от другого интерфейса
396	Ожидается интерфейс
397	Интерфейсы поддерживают только публичное наследование

© 2000-2025, MetaQuotes Ltd.

Номер	Описание
398	Интерфейс не может содержать члены
399	Нельзя создавать объекты интерфейса напрямую, только через наследование
400	Спецификатор нельзя использовать в форвардном объявлении
401	Наследование от данного класса невозможно, так как он объявлен со спецификатором final
402	Нельзя переопределить метод, объявленный со спецификатором final
403	Спецификатор final можно применять только к виртуальным функциям
404	Метод, помеченный спецификатором override, на самом деле не перекрывает никакую функцию базового класса
405	Спецификатор не разрешен в определении функции, только в объявлении
406	Невозможно привести данный тип к указанному
407	Данный тип нельзя использовать для ресурсной переменной
408	Ошибка в файле проекта
409	Не может быть использован в качестве члена union
410	Неоднозначность выбора для данного имени, необходимо явно определить контекст использования
411	Нельзя использовать данную структуру из DLL
412	Нельзя вызывать функцию, помеченную спецификатором delete
413	MQL4 не поддерживается. Для компиляции этой программы используйте MetaEditor, который находится в папке установке вашего терминала MetaTrader 4

© 2000-2025, MetaQuotes Ltd.

Ошибки времени выполнения

GetLastError() - функция, возвращающая код последней ошибки, которая хранится в предопределенной переменной _LastError. Значение этой переменной можно сбросить в ноль функцией ResetLastError().

Константа	Значение	Описание
ERR_SUCCESS	0	Операция выполнена успешно
ERR_INTERNAL_ERROR	4001	Неожиданная внутренняя ошибка
ERR_WRONG_INTERNAL_PARAMETER	4002	Ошибочный параметр при внутреннем вызове функции клиентского терминала
ERR_INVALID_PARAMETER	4003	Ошибочный параметр при вызове системной функции
ERR_NOT_ENOUGH_MEMORY	4004	Недостаточно памяти для выполнения системной функции
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	Структура содержит объекты строк и/или динамических массивов и/или структуры с такими объектами и/или классы
ERR_INVALID_ARRAY	4006	Массив неподходящего типа, неподходящего размера или испорченный объект динамического массива
ERR_ARRAY_RESIZE_ERROR	4007	Недостаточно памяти для перераспределения массива либо попытка изменения размера статического массива
ERR_STRING_RESIZE_ERROR	4008	Недостаточно памяти для перераспределения строки
ERR_NOTINITIALIZED_STRING	4009	Неинициализированная строка
ERR_INVALID_DATETIME	4010	Неправильное значение даты и/или времени
ERR_ARRAY_BAD_SIZE	4011	Общее число элементов в массиве не может превышать 2147483647
ERR_INVALID_POINTER	4012	Ошибочный указатель
ERR_INVALID_POINTER_TYPE	4013	Ошибочный тип указателя
ERR_FUNCTION_NOT_ALLOWED	4014	Системная функция не разрешена для вызова

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_RESOURCE_NAME_DUPLICATED	4015	Совпадение имени динамического и статического ресурсов
ERR_RESOURCE_NOT_FOUND	4016	Ресурс с таким именем в EX5 не найден
ERR_RESOURCE_UNSUPPORTED_TYPE	4017	Неподдерживаемый тип ресурса или размер более 16 MB
ERR_RESOURCE_NAME_IS_TOO_LONG	4018	Имя ресурса превышает 63 символа
ERR_MATH_OVERFLOW	4019	При вычислении математической функции произошло переполнение
ERR_SLEEP_ERROR	4020	Выход за дату окончания тестирования после вызова Sleep()
ERR_PROGRAM_STOPPED	4022	Тестирование было прекращено принудительно извне. Например, прервана оптимизацию, или закрыто окно визуального тестирования, или остановлен агент тестирования
ERR_INVALID_TYPE	4023	Неподходящий тип
ERR_INVALID_HANDLE	4024	Невалидный хендл
ERR_TOO_MANY_OBJECTS	4025	Пул объектов заполнен
Графики		
ERR_CHART_WRONG_ID	4101	Ошибочный идентификатор графика
ERR_CHART_NO_REPLY	4102	График не отвечает
ERR_CHART_NOT_FOUND	4103	График не найден
ERR_CHART_NO_EXPERT	4104	У графика нет эксперта, который мог бы обработать событие
ERR_CHART_CANNOT_OPEN	4105	Ошибка открытия графика
ERR_CHART_CANNOT_CHANGE	4106	Ошибка при изменении для графика символа и периода
ERR_CHART_WRONG_PARAMETER	4107	Ошибочное значение параметра для функции по работе с графиком

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_CHART_CANNOT_CREATE_TIMER	4108	Ошибка при создании таймера
ERR_CHART_WRONG_PROPERTY	4109	Ошибочный идентификатор свойства графика
ERR_CHART_SCREENSHOT_FAILED	4110	Ошибка при создании скриншота
ERR_CHART_NAVIGATE_FAILED	4111	Ошибка навигации по графику
ERR_CHART_TEMPLATE_FAILED	4112	Ошибка при применении шаблона
ERR_CHART_WINDOW_NOT_FOUND	4113	Подокно, содержащее указанный индикатор, не найдено
ERR_CHART_INDICATOR_CANNOT_ADD	4114	Ошибка при добавлении индикатора на график
ERR_CHART_INDICATOR_CANNOT_DEL	4115	Ошибка при удалении индикатора с графика
ERR_CHART_INDICATOR_NOT_FOUND	4116	Индикатор не найден на указанном графике
Графические объекты		
ERR_OBJECT_ERROR	4201	Ошибка при работе с графическим объектом
ERR_OBJECT_NOT_FOUND	4202	Графический объект не найден
ERR_OBJECT_WRONG_PROPERTY	4203	Ошибочный идентификатор свойства графического объекта
ERR_OBJECT_GETDATE_FAILED	4204	Невозможно получить дату, соответствующую значению
ERR_OBJECT_GETVALUE_FAILED	4205	Невозможно получить значение, соответствующее дате
MarketInfo		
ERR_MARKET_UNKNOWN_SYMBOL	4301	Неизвестный символ
ERR_MARKET_NOT_SELECTED	4302	Символ не выбран в MarketWatch
ERR_MARKET_WRONG_PROPERTY	4303	Ошибочный идентификатор свойства символа
ERR_MARKET_LASTTIME_UNKNOWN	4304	Время последнего тика неизвестно (тиков не было)
ERR_MARKET_SELECT_ERROR	4305	Ошибка добавления или удаления символа в MarketWatch

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_MARKET_SELECT_LIMIT	4306	Превышен лимит выбранных символов в MarketWatch
ERR_MARKET_SESSION_INDEX	4307	Неправильный индекс сессии при вызове функции SymbolInfoSessionQuote/SymbolInfoSessionTrade
Доступ к истории		
ERR_HISTORY_NOT_FOUND	4401	Запрашиваемая история не найдена
ERR_HISTORY_WRONG_PROPERTY	4402	Ошибочный идентификатор свойства истории
ERR_HISTORY_TIMEOUT	4403	Превышен таймаут при запросе истории
ERR_HISTORY_BARS_LIMIT	4404	Количество запрашиваемых баров ограничено настройками терминала
ERR_HISTORY_LOAD_ERRORS	4405	Множество ошибок при загрузке истории
ERR_HISTORY_SMALL_BUFFER	4407	Принимающий массив слишком мал чтобы вместить все запрошенные данные
Глобальные переменные терминала		
ERR_GLOBALVARIABLE_NOT_FOUND	4501	Глобальная переменная клиентского терминала не найдена
ERR_GLOBALVARIABLE_EXISTS	4502	Глобальная переменная клиентского терминала с таким именем уже существует
ERR_GLOBALVARIABLE_NOT_MODIFIED	4503	Не было модификаций глобальных переменных
ERR_GLOBALVARIABLE_CANNOTREAD	4504	Не удалось открыть и прочитать файл со значениями глобальных переменных
ERR_GLOBALVARIABLE_CANNOTWRITE	4505	Не удалось записать файл со значениями глобальных переменных
ERR_MAIL_SEND_FAILED	4510	Не удалось отправить письмо
ERR_PLAY_SOUND_FAILED	4511	Не удалось воспроизвести звук

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_MQL5_WRONG_PROPERTY	4512	Ошибочный идентификатор свойства программы
ERR_TERMINAL_WRONG_PROPERTY	4513	Ошибочный идентификатор свойства терминала
ERR_FTP_SEND_FAILED	4514	Не удалось отправить файл по ftp
ERR_NOTIFICATION_SEND_FAILED	4515	Не удалось отправить уведомление
ERR_NOTIFICATION_WRONG_PARAMETER	4516	Неверный параметр для отправки уведомления - в функцию SendNotification() передали пустую строку или NULL
ERR_NOTIFICATION_WRONG_SETTINGS	4517	Неверные настройки уведомлений в терминале (не указан ID или не выставлено разрешение)
ERR_NOTIFICATION_TOO_FREQUENT	4518	Слишком частая отправка уведомлений
ERR_FTP_NOSERVER	4519	Не указан FTP сервер
ERR_FTP_NOLOGIN	4520	Не указан FTP логин
ERR_FTP_FILE_ERROR	4521	Не найден файл в директории MQL5\Files для отправки на FTP сервер
ERR_FTP_CONNECT_FAILED	4522	Ошибка при подключении к FTP серверу
ERR_FTP_CHANGEDIR	4523	На FTP сервере не найдена директория для выгрузки файла
Буферы пользовательских индикаторов		
ERR_BUFFERS_NO_MEMORY	4601	Недостаточно памяти для распределения индикаторных буферов
ERR_BUFFERS_WRONG_INDEX	4602	Ошибочный индекс своего индикаторного буфера
Свойства пользовательских индикаторов		
ERR_CUSTOM_WRONG_PROPERTY	4603	Ошибочный идентификатор свойства пользовательского индикатора

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
Account		
ERR_ACCOUNT_WRONG_PROPERTY	4701	Ошибочный идентификатор свойства счета
ERR_TRADE_WRONG_PROPERTY	4751	Ошибочный идентификатор свойства торговли
ERR_TRADE_DISABLED	4752	Торговля для эксперта запрещена
ERR_TRADE_POSITION_NOT_FOUND	4753	Позиция не найдена
ERR_TRADE_ORDER_NOT_FOUND	4754	Ордер не найден
ERR_TRADE_DEAL_NOT_FOUND	4755	Сделка не найдена
ERR_TRADE_SEND_FAILED	4756	Не удалось отправить торговый запрос
ERR_TRADE_CALC_FAILED	4758	Не удалось вычислить значение прибыли или маржи
Индикаторы		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	Неизвестный символ
ERR_INDICATOR_CANNOT_CREATE	4802	Индикатор не может быть создан
ERR_INDICATOR_NO_MEMORY	4803	Недостаточно памяти для добавления индикатора
ERR_INDICATOR_CANNOT_APPLY	4804	Индикатор не может быть применен к другому индикатору
ERR_INDICATOR_CANNOT_ADD	4805	Ошибка при добавлении индикатора
ERR_INDICATOR_DATA_NOT_FOUND	4806	Запрошенные данные не найдены
ERR_INDICATOR_WRONG_HANDLE	4807	Ошибочный хэндл индикатора
ERR_INDICATOR_WRONG_PARAMETERS	4808	Неправильное количество параметров при создании индикатора
ERR_INDICATOR_PARAMETERS_MISSING	4809	Отсутствуют параметры при создании индикатора
ERR_INDICATOR_CUSTOM_NAME	4810	Первым параметром в массиве должно быть имя пользовательского индикатора
ERR_INDICATOR_PARAMETER_TYPE	4811	Неправильный тип параметра в массиве при создании

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
		индикатора
ERR_INDICATOR_WRONG_INDEX	4812	Ошибочный индекс запрашиваемого индикаторного буфера
Стакан цен		
ERR_BOOKS_CANNOT_ADD	4901	Стакан цен не может быть добавлен
ERR_BOOKS_CANNOT_DELETE	4902	Стакан цен не может быть удален
ERR_BOOKS_CANNOT_GET	4903	Данные стакана цен не могут быть получены
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	Ошибка при подписке на получение новых данных стакана цен
Файловые операции		
ERR_TOO_MANY_FILES	5001	Не может быть открыто одновременно более 64 файлов
ERR_WRONG_FILENAME	5002	Недопустимое имя файла
ERR_TOO_LONG_FILENAME	5003	Слишком длинное имя файла
ERR_CANNOT_OPEN_FILE	5004	Ошибка открытия файла
ERR_FILE_CACHEBUFFER_ERROR	5005	Недостаточно памяти для кеша чтения
ERR_CANNOT_DELETE_FILE	5006	Ошибка удаления файла
ERR_INVALID_FILEHANDLE	5007	Файл с таким хэндлом уже был закрыт, либо не открывался вообще
ERR_WRONG_FILEHANDLE	5008	Ошибочный хэндл файла
ERR_FILE_NOTTOWRITE	5009	Файл должен быть открыт для записи
ERR_FILE_NOTTOREAD	5010	Файл должен быть открыт для чтения
ERR_FILE_NOTBIN	5011	Файл должен быть открыт как бинарный
ERR_FILE_NOTTXT	5012	Файл должен быть открыт как текстовый
ERR_FILE_NOTTXTORCSV	5013	Файл должен быть открыт как текстовый или CSV

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_FILE_NOTCSV	5014	Файл должен быть открыт как CSV
ERR_FILE_READERROR	5015	Ошибка чтения файла
ERR_FILE_BINSTRINGSIZE	5016	Должен быть указан размер строки, так как файл открыт как бинарный
ERR_INCOMPATIBLE_FILE	5017	Для строковых массивов должен быть текстовый файл, для остальных - бинарный
ERR_FILE_IS_DIRECTORY	5018	Это не файл, а директория
ERR_FILE_NOT_EXIST	5019	Файл не существует
ERR_FILE_CANNOT_REWRITE	5020	Файл не может быть переписан
ERR_WRONG_DIRECTORYNAME	5021	Ошибочное имя директории
ERR_DIRECTORY_NOT_EXIST	5022	Директория не существует
ERR_FILE_ISNOT_DIRECTORY	5023	Это файл, а не директория
ERR_CANNOT_DELETE_DIRECTORY	5024	Директория не может быть удалена
ERR_CANNOT_CLEAN_DIRECTORY	5025	Не удалось очистить директорию (возможно, один или несколько файлов заблокированы и операция удаления не удалась)
ERR_FILE_WRITEERROR	5026	Не удалось записать ресурс в файл
ERR_FILE_ENDOFFILE	5027	Не удалось прочитать следующую порцию данных из CSV-файла (FileReadString, FileReadNumber, FileReadDatetime, FileReadBool), так как достигнут конец файла
Преобразование строк		
ERR_NO_STRING_DATE	5030	В строке нет даты
ERR_WRONG_STRING_DATE	5031	В строке ошибочная дата
ERR_WRONG_STRING_TIME	5032	В строке ошибочное время
ERR_STRING_TIME_ERROR	5033	Ошибка преобразования строки в дату
ERR_STRING_OUT_OF_MEMORY	5034	Недостаточно памяти для строки

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_STRING_SMALL_LEN	5035	Длина строки меньше, чем ожидалось
ERR_STRING_TOO_BIGNUMBER	5036	Слишком большое число, больше, чем ULONG_MAX
ERR_WRONG_FORMATSTRING	5037	Ошибочная форматная строка
ERR_TOO_MANY_FORMATTERS	5038	Форматных спецификаторов больше, чем параметров
ERR_TOO_MANY_PARAMETERS	5039	Параметров больше, чем форматных спецификаторов
ERR_WRONG_STRING_PARAMETER	5040	Испорченный параметр типа string
ERR_STRINGPOS_OUTOFRANGE	5041	Позиция за пределами строки
ERR_STRING_ZEROADDED	5042	К концу строки добавлен 0, бесполезная операция
ERR_STRING_UNKNOWNTYPE	5043	Неизвестный тип данных при конвертации в строку
ERR_WRONG_STRING_OBJECT	5044	Испорченный объект строки
Работа с массивами		
ERR_INCOMPATIBLE_ARRAYS	5050	Копирование несовместимых массивов. Строковый массив может быть скопирован только в строковый, а числовой массив - в числовой
ERR_SMALL_ASSERIES_ARRAY	5051	Приемный массив объявлен как AS_SERIES, и он недостаточного размера
ERR_SMALL_ARRAY	5052	Слишком маленький массив, стартовая позиция за пределами массива
ERR_ZEROSIZE_ARRAY	5053	Массив нулевой длины
ERR_NUMBER_ARRAYS_ONLY	5054	Должен быть числовой массив
ERR_ONEDIM_ARRAYS_ONLY	5055	Должен быть одномерный массив
ERR_SERIES_ARRAY	5056	Таймсерия не может быть использована
ERR_DOUBLE_ARRAY_ONLY	5057	Должен быть массив типа double
ERR_FLOAT_ARRAY_ONLY	5058	Должен быть массив типа float

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_LONG_ARRAY_ONLY	5059	Должен быть массив типа long
ERR_INT_ARRAY_ONLY	5060	Должен быть массив типа int
ERR_SHORT_ARRAY_ONLY	5061	Должен быть массив типа short
ERR_CHAR_ARRAY_ONLY	5062	Должен быть массив типа char
ERR_STRING_ARRAY_ONLY	5063	Должен быть массив типа string
Работа с OpenCL		
ERR_OPENCL_NOT_SUPPORTED	5100	Функции OpenCL на данном компьютере не поддерживаются
ERR_OPENCL_INTERNAL	5101	Внутренняя ошибка при выполнении OpenCL
ERR_OPENCL_INVALID_HANDLE	5102	Неправильный хэндл OpenCL
ERR_OPENCL_CONTEXT_CREATE	5103	Ошибка при создании контекста OpenCL
ERR_OPENCL_QUEUE_CREATE	5104	Ошибка создания очереди выполнения в OpenCL
ERR_OPENCL_PROGRAM_CREATE	5105	Ошибка при компиляции программы OpenCL
ERR_OPENCL_TOO_LONG_KERNEL_NAME	5106	Слишком длинное имя точки входа (кернел OpenCL)
ERR_OPENCL_KERNEL_CREATE	5107	Ошибка создания кернел - точки входа OpenCL
ERR_OPENCL_SET_KERNEL_PARAMETER	5108	Ошибка при установке параметров для кернел OpenCL (точки входа в программу OpenCL)
ERR_OPENCL_EXECUTE	5109	Ошибка выполнения программы OpenCL
ERR_OPENCL_WRONG_BUFFER_SIZE	5110	Неверный размер буфера OpenCL
ERR_OPENCL_WRONG_BUFFER_OFFSET	5111	Неверное смещение в буфере OpenCL
ERR_OPENCL_BUFFER_CREATE	5112	Ошибка создания буфера OpenCL
ERR_OPENCL_TOO_MANY_OBJECTS	5113	Превышено максимальное число OpenCL объектов
ERR_OPENCL_SELECTDEVICE	5114	Ошибка выбора OpenCL устройства

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
Работа с базами данных		
ERR_DATABASE_INTERNAL	5120	Внутренняя ошибка базы данных
ERR_DATABASE_INVALID_HANDLE	5121	Невалидный хендл базы данных
ERR_DATABASE_TOO_MANY_OBJECTS	5122	Превышено максимально допустимое количество объектов Database
ERR_DATABASE_CONNECT	5123	Ошибка подключения к базе данных
ERR_DATABASE_EXECUTE	5124	Ошибка выполнения запроса
ERR_DATABASE_PREPARE	5125	Ошибка создания запроса
ERR_DATABASE_NO_MORE_DATA	5126	Данных для чтения больше нет
ERR_DATABASE_STEP	5127	Ошибка перехода к следующей записи запроса
ERR_DATABASE_NOT_READY	5128	Данные для чтения результатов запроса еще не готовы
ERR_DATABASE_BIND_PARAMETERS	5129	Ошибка автоподстановки параметров в SQL-запрос
ERR_DATABASE_QUERY_NOT_READONLY	5130	
Работа с WebRequest()		
ERR_WEBREQUEST_INVALID_ADDRESS	5200	URL не прошел проверку
ERR_WEBREQUEST_CONNECT_FAILED	5201	Не удалось подключиться к указанному URL
ERR_WEBREQUEST_TIMEOUT	5202	Превышен таймаут получения данных
ERR_WEBREQUEST_REQUEST_FAILED	5203	Ошибка в результате выполнения HTTP запроса
Работа с сетью (сокетами)		
ERR_NETSOCKET_INVALIDHANDLE	5270	В функцию передан неверный хэндл сокета
ERR_NETSOCKET_TOO_MANY_OPENED	5271	Открыто слишком много сокетов (максимум 128)
ERR_NETSOCKET_CANNOT_CONNECT	5272	Ошибка соединения с удаленным хостом
ERR_NETSOCKET_IO_ERROR	5273	Ошибка отправки/получения данных из сокета

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_NETSOCKET_HANDSHAKE_FAILED	5274	Ошибка установления защищенного соединения (TLS Handshake)
ERR_NETSOCKET_NO_CERTIFICATE	5275	Отсутствуют данные о сертификате, которым защищено подключение
Пользовательские символы		
ERR_NOT_CUSTOM_SYMBOL	5300	Должен быть указан пользовательский символ
ERR_CUSTOM_SYMBOL_WRONG_NAME	5301	Некорректное имя пользовательского символа. В имени символа можно использовать только латинские буквы без знаков препинания, пробелов и спецсимволов (допускаются ".", "_", "&" и "#"). Не рекомендуется использовать символы <, >, :, ", /,\, |, ?, *.
ERR_CUSTOM_SYMBOL_NAME_LONG	5302	Слишком длинное имя для пользовательского символа. Длина имени символа не должна превышать 32 знака с учётом завершающего 0
ERR_CUSTOM_SYMBOL_PATH_LONG	5303	Слишком длинный путь для пользовательского символа. Длина пути не более 128 знаков с учётом "Custom\\", имени символа, разделителей групп и завершающего 0
ERR_CUSTOM_SYMBOL_EXIST	5304	Пользовательский символ с таким именем уже существует
ERR_CUSTOM_SYMBOL_ERROR	5305	Ошибка при создании, удалении или изменении пользовательского символа
ERR_CUSTOM_SYMBOL_SELECTED	5306	Попытка удалить пользовательский символ, выбранный в обзоре рынка (Market Watch)
ERR_CUSTOM_SYMBOL_PROPERTY_WRONG	5307	Неправильное свойство пользовательского символа
ERR_CUSTOM_SYMBOL_PARAMETER_ERROR	5308	Ошибочный параметр при установке свойства

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
		пользовательского символа
ERR_CUSTOM_SYMBOL_PARAMETER_LONG	5309	Слишком длинный строковый параметр при установке свойства пользовательского символа
ERR_CUSTOM_TICKS_WRONG_ORDER	5310	Не упорядоченный по времени массив тиков
Экономический календарь		
ERR_CALENDAR_MORE_DATA	5400	Размер массива недостаточен для получения описаний всех значений
ERR_CALENDAR_TIMEOUT	5401	Превышен лимит запроса по времени
ERR_CALENDAR_NO_DATA	5402	Страна не найдена
Работа с базами данных		
ERR_DATABASE_ERROR	5601	Общая ошибка
ERR_DATABASE_LOGIC	5602	Внутренняя логическая ошибка в SQLite
ERR_DATABASE_PERM	5603	Отказано в доступе
ERR_DATABASE_ABORT	5604	Процедура обратного вызова запросила прерывание
ERR_DATABASE_BUSY	5605	Файл базы данных заблокирован
ERR_DATABASE_LOCKED	5606	Таблица в базе данных заблокирована
ERR_DATABASE_NOMEM	5607	Сбой malloc ()
ERR_DATABASE_READONLY	5608	Попытка записи в базу данных, доступной только для чтения
ERR_DATABASE_INTERRUPT	5609	Операция прекращена с помощью sqlite3_interrupt ()
ERR_DATABASE_IOERR	5610	Ошибка дискового ввода-вывода
ERR_DATABASE_CORRUPT	5611	Образ диска базы данных испорчен
ERR_DATABASE_NOTFOUND	5612	Неизвестный код операции в sqlite3_file_control ()
ERR_DATABASE_FULL	5613	Ошибка вставки, так как база данных заполнена

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_DATABASE_CANTOPEN	5614	Невозможно открыть файл базы данных
ERR_DATABASE_PROTOCOL	5615	Ошибка протокола блокировки базы данных
ERR_DATABASE_EMPTY	5616	Только для внутреннего использования
ERR_DATABASE_SCHEMA	5617	Схема базы данных изменена
ERR_DATABASE_TOOBIG	5618	Строка или BLOB превышает ограничение по размеру
ERR_DATABASE_CONSTRAINT	5619	Прервано из-за нарушения ограничения
ERR_DATABASE_MISMATCH	5620	Несоответствие типов данных
ERR_DATABASE_MISUSE	5621	Ошибка неправильного использования библиотеки
ERR_DATABASE_NOLFS	5622	Использование функций операционной системы, не поддерживаемых на хосте
ERR_DATABASE_AUTH	5623	Отказано в авторизации
ERR_DATABASE_FORMAT	5624	Не используется
ERR_DATABASE_RANGE	5625	2-й параметр для sqlite3_bind находится вне диапазона
ERR_DATABASE_NOTADB	5626	Открытый файл не является файлом базы данных
Методы матриц и векторов		
ERR_MATRIX_INTERNAL	5700	Внутренняя ошибка исполняющей подсистемы матриц/векторов
ERR_MATRIX_NOT_INITIALIZED	5701	Матрица/вектор не инициализирован
ERR_MATRIX_INCONSISTENT	5702	Несогласованный размер матриц/векторов в операции
ERR_MATRIX_INVALID_SIZE	5703	Некорректный размер матрицы/вектора
ERR_MATRIX_INVALID_TYPE	5704	Некорректный тип матрицы/вектора
ERR_MATRIX_FUNC_NOT_ALLOWED	5705	Функция недоступна для данной матрицы/вектора

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
ERR_MATRIX_CONTAINS_NAN	5706	Матрица/вектор содержит нечисла (Nan/Inf)
ONNX модели		
ERR_ONNX_INTERNAL	5800	Внутренняя ошибка ONNX стандарта
ERR_ONNX_NOT_INITIALIZED	5801	Ошибка инициализации ONNX Runtime API
ERR_ONNX_NOT_SUPPORTED	5802	Свойство или значение неподдерживаются языком MQL5
ERR_ONNX_RUN_FAILED	5803	Ошибка запуска ONNX runtime API
ERR_ONNX_INVALID_PARAMETERS_COUNT	5804	В OnnxRun передано неверное количество параметров
ERR_ONNX_INVALID_PARAMETER	5805	Некорректное значение параметра
ERR_ONNX_INVALID_PARAMETER_TYPE	5806	Некорректный тип параметра
ERR_ONNX_INVALID_PARAMETER_SIZE	5807	Некорректный размер параметра
ERR_ONNX_WRONG_DIMENSION	5808	Размерность тензора не задана или указана неверно
Пользовательские ошибки		
ERR_USER_ERROR_FIRST	65536	С этого кода начинаются ошибки, задаваемые пользователем

Смотри также

Коды возврата торгового сервера

© 2000-2025, MetaQuotes Ltd.

Константы ввода/вывода

Константы:

- Флаги открытия файлов
- Свойства файлов
- Позиционирование внутри файла
- Использование кодовой страницы
- MessageBox

© 2000-2025, MetaQuotes Ltd.

Флаги открытия файлов

Значения флагов, определяющих режим работы с файлом. Флаги определены следующим образом:

Идентификатор	Значение	Описание
FILE_READ	1	Файл открывается для чтения. Флаг используется при открытии файлов (FileOpen()). При открытии файла обязательно должен быть указан флаг FILE_WRITE и/или флаг FILE_READ
FILE_WRITE	2	Файл открывается для записи. Флаг используется при открытии файлов (FileOpen()). При открытии файла обязательно должен быть указан флаг FILE_WRITE и/или флаг FILE_READ
FILE_BIN	4	Двоичный режим чтения-записи (без преобразования из строки и в строку). Флаг используется при открытии файлов (FileOpen())
FILE_CSV	8	Файл типа csv (все записанные элементы преобразуются к строкам соответствующего типа, unicode или ansi, и разделяются разделителем). Флаг используется при открытии файлов (FileOpen())
FILE_TXT	16	Простой текстовый файл (тот же csv, однако разделитель не принимается во внимание). Флаг используется при открытии файлов (FileOpen())
FILE_ANSI	32	Строки типа ANSI (однобайтовые символы). Флаг используется при открытии файлов (FileOpen())
FILE_UNICODE	64	Строки типа UNICODE (двухбайтовые символы). Флаг используется при открытии файлов (FileOpen())
FILE_SHARE_READ	128	Совместный доступ по чтению со стороны нескольких программ. Флаг используется при открытии файлов (FileOpen()), но не заменяет при открытии файла необходимости указать FILE_WRITE и/или флаг FILE_READ
FILE_SHARE_WRITE	256	Совместный доступ по записи со стороны нескольких программ. Флаг используется при открытии файлов (FileOpen()), но не заменяет при открытии файла необходимости указать FILE_WRITE и/или флаг FILE_READ
FILE_REWRITE	512	Возможность перезаписывания файла функциями FileCopy() и FileMove(). Файл должен существовать или открываться для записи. В противном случае файл открыт не будет
FILE_COMMON	4096	Расположение файла в общей папке всех клиентских терминалов \Terminal\Common\Files. Флаг

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение	Описание
		используется при открытии файлов (FileOpen()), копировании файлов (FileCopy(), FileMove()) и проверке существования файлов (FileIsExist())

При открытии файла можно указать один или более флагов, такое сочетание называется комбинацией флагов. Комбинация флагов записывается с помощью знака операции логического ИЛИ (|), который ставится между перечисляемыми флагами. Например, чтобы открыть файл в формате CSV одновременно на чтение и на запись, можно указать комбинацию FILE_READ|FILE_WRITE|FILE_CSV.

Пример:

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

Есть некоторые особенности работы при указании флагов чтения и записи:

- Если указан FILE_READ - делается попытка открытия уже существующего файла. Если файл не существует, то открыть файл не получится, новый файл не создается.
- Если FILE_READ|FILE_WRITE - создаётся новый файл если файл с таким именем отсутствует.
- Если FILE_WRITE - файл создается заново с нулевым размером.

При открытии файла обязательно должен быть указан флаг FILE_WRITE и/или флаг FILE_READ.

Флаги, определяющие тип чтения открытого файла, имеют приоритет. Самый старший флаг FILE_CSV, затем следует по старшинству флаг FILE_BIN, и наименьший приоритет имеет флаг FILE_TXT. Таким образом, если вдруг будут указаны сразу несколько флагов (FILE_TXT|FILE_CSV или FILE_TXT|FILE_BIN или FILE_BIN|FILE_CSV), то будет использован самый старший по приоритету флаг.

Флаги, определяющие тип кодировки, также имеют приоритет. Флаг FILE_UNICODE имеет более старший приоритет,чем флаг FILE_ANSI. Поэтому при указании комбинации FILE_UNICODE|FILE_ANSI будет использоваться флаг FILE_UNICODE.

Если не указано ни FILE_UNICODE, ни FILE_ANSI, то подразумевается FILE_UNICODE. Если не указано ни FILE_CSV, ни FILE_BIN, ни FILE_TXT, то подразумевается FILE_CSV.

Если файл открыт на чтение как текстовый (FILE_TXT или FILE_CSV), и при этом в начале файла будет обнаружен специальный двухбайтовый признак 0xff,0xfe, то флаг кодировки будет FILE_UNICODE, даже если указан флаг FILE_ANSI.

Смотри также

Файловые операции

© 2000-2025, MetaQuotes Ltd.

Свойства файлов

Для получения свойств файлов используется функция FileGetInteger(), которой при вызове передается идентификатор требуемого свойства из перечисления ENUM_FILE_PROPERTY_INTEGER

ENUM_FILE_PROPERTY_INTEGER

Идентификатор	Описание идентификатора
FILE_EXISTS	Проверка на существование
FILE_CREATE_DATE	Дата создания
FILE_MODIFY_DATE	Дата последнего изменения
FILE_ACCESS_DATE	Дата последнего доступа к файлу
FILE_SIZE	Размер файла в байтах
FILE_POSITION	Позиция указателя в файле
FILE_END	Получение признака конца файла
FILE_LINE_END	Получение признака конца строки
FILE_IS_COMMON	Файл открыт в общей папке всех клиентских терминалов (смотри FILE_COMMON)
FILE_IS_TEXT	Файл открыт как текстовый (смотри FILE_TXT)
FILE_IS_BINARY	Файл открыт как бинарный (смотри FILE_BIN)
FILE_IS_CSV	Файл открыт как CSV (смотри FILE_CSV)
FILE_IS_ANSI	Файл открыт как ANSI (смотри FILE_ANSI)
FILE_IS_READABLE	Файл открыт с возможностью чтения (смотри FILE_READ)
FILE_IS_WRITABLE	Файл открыт с возможностью записи (смотри FILE_WRITE)

Функция FileGetInteger() имеет два варианта вызова. В первом варианте для получения свойств файла указывается его хэндл, полученный при открытии файла функцией FileOpen(). Этот вариант позволяет получить все свойства файла.

Второй вариант функции FileGetInteger() возвращает значения свойств файла по его имени. В этом варианте можно получить только следующие общие свойства:

- FILE_EXISTS - существование указанного по имени файла;
- FILE_CREATE_DATE - дата создания файла с указанным именем;
- FILE_MODIFY_DATE - дата изменения файла с указанным именем;
- FILE_ACCESS_DATE - дата последнего доступа к файлу с указанным именем;
- FILE_SIZE - размер файла с указанным именем.

При попытке получения других свойств, кроме указанных выше, второй вариант вызова функции FileGetInteger() вернет ошибку.

© 2000-2025, MetaQuotes Ltd.

Позиционирование внутри файла

Большая часть файловых функций связана с операциями чтения/записи информации. При этом с помощью функции FileSeek() можно указывать положение файлового указателя на позицию внутри файла, с которой будет производится следующая операция чтения или записи. Перечисление ENUM_FILE_POSITION содержит допустимые положения указателя, относительно которого можно указать смещения в байтах для следующей операции.

ENUM_FILE_POSITION

Идентификатор	Описание
SEEK_SET	Начало файла
SEEK_CUR	Текущая позиция файлового указателя
SEEK_END	Конец файла

Смотри также

FileIsEnding, FileIsLineEnding

© 2000-2025, MetaQuotes Ltd.

Использование кодовой страницы в операциях преобразования строк

При операциях конвертации строковых переменных в массивы типа char и обратно в языке MQL5 используется кодировка, соответствующая по умолчанию текущей ANSI кодировке операционной системы Windows (CP_ACP). Если требуется указать иной тип кодировки, то его можно задать дополнительным параметром для функций CharArrayToString(), StringToCharArray() и FileOpen().

В таблице приведены встроенные константы для некоторых наиболее востребованных кодовых страниц. Неперечисленные кодовые страницы можно указать кодом, соответствующим этой странице.

Встроенные константы кодовых страниц

Константа	Значение	Описание
CP_ACP	0	Текущая кодовая страница ANSI кодировка в операционной системе Windows
CP_OEMCP	1	Текущая кодовая страница OEM.
CP_MACCP	2	Текущая кодовая страница Macintosh. **Примечание:** Это значение преимущественно используется в ранее созданных программных кодах и теперь в нем нет необходимости, так как современные компьютеры Macintosh используют Unicode кодировку.
CP_THREAD_ACP	3	Кодировка Windows ANSI для текущего потока выполнения.
CP_SYMBOL	42	Кодовая страница Symbol
CP_UTF7	65000	Кодовая страница UTF-7.
CP_UTF8	65001	Кодовая страница UTF-8.

Смотри также

Состояние клиентского терминала

© 2000-2025, MetaQuotes Ltd.

Константы диалогового окна MessageBox

Коды возврата функции MessageBox(). Если окно сообщения имеет кнопку Отмена (Cancel), то функция возвращает значение IDCANCEL при нажатой клавише ESC или кнопке Отмена (Cancel). Если окно сообщения не имеет кнопки Отмена (Cancel), нажатие ESC не дает никакого эффекта.

Константа	Значение	Описание
IDOK	1	Выбрана кнопка OK
IDCANCEL	2	Выбрана кнопка Отмена (Cancel)
IDABORT	3	Выбрана кнопка Прервать (Abort)
IDRETRY	4	Выбрана кнопка Повтор (Retry)
IDIGNORE	5	Выбрана кнопка Пропустить (Ignore)
IDYES	6	Выбрана кнопка Да (Yes)
IDNO	7	Выбрана кнопка Нет (No)
IDTRYAGAIN	10	Выбрана кнопка Повторить (Try Again)
IDCONTINUE	11	Выбрана кнопка Продолжить (Continue)

Основные флаги функции MessageBox() определяют содержание и поведение диалогового окна. Это значение может быть комбинацией флагов из следующих групп флагов:

Константа	Значение	Описание
MB_OK	0x00000000	Окно сообщения содержит одну кнопку: OK. По умолчанию
MB_OKCANCEL	0x00000001	Окно сообщения содержит две кнопки: OK и Cancel
MB_ABORTRETRYIGNORE	0x00000002	Окно сообщения содержит три кнопки: Abort, Retry и Ignore
MB_YESNOCANCEL	0x00000003	Окно сообщения содержит три кнопки: Yes, No и Cancel
MB_YESNO	0x00000004	Окно сообщения содержит две кнопки: Yes и No
MB_RETRYCANCEL	0x00000005	Окно сообщения содержит две кнопки: Retry и Cancel
MB_CANCELTRYCONTINUE	0x00000006	Окно сообщения содержит три кнопки: Cancel, Try Again, Continue

Для отображения иконки в окне сообщения необходимо определить дополнительные флаги:

© 2000-2025, MetaQuotes Ltd.

Константа	Значение	Описание
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	Изображение знака STOP
MB_ICONQUESTION	0x00000020	Изображение вопросительного знака
MB_ICONEXCLAMATION, MB_ICONWARNING	0x00000030	Изображение восклицательного знака
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	Изображение, состоящее из строчного знака **i** в круге

Кнопки по умолчанию задаются следующими флагами:

Константа	Значение	Описание
MB_DEFBUTTON1	0x00000000	Первая кнопка MB_DEFBUTTON1 - кнопка выбрана по умолчанию, если MB_DEFBUTTON2, MB_DEFBUTTON3, или MB_DEFBUTTON4 не определены
MB_DEFBUTTON2	0x00000100	Вторая кнопка - кнопка по умолчанию
MB_DEFBUTTON3	0x00000200	Третья кнопка - кнопка по умолчанию
MB_DEFBUTTON4	0x00000300	Четвертая кнопка - кнопка по умолчанию

© 2000-2025, MetaQuotes Ltd.

Программы MQL5

Для того чтобы mql5-программа могла работать, она должна быть скомпилирована (кнопка "Компилировать" или клавиша F7). Компиляция должна пройти без ошибок (допускаются предупреждения, которые необходимо проанализировать). При этом в соответствующей директории *terminal_dir\MQL5\Experts*, *terminal_dir\MQL5\indicators* или *terminal_dir\MQL5\scripts* должен быть создан выполняемый файл с тем же именем и расширением EX5. Именно этот файл может быть запущен на выполнение.

Особенности работы mql5-программ описаны в разделах:

- Выполнение программ – порядок вызова предопределенных функций-обработчиков событий;
- Тестирование торговых стратегий – особенности работы программ в тестере стратегий;
- События клиентского терминала – описание событий, которые можно обрабатывать в программах;
- Вызов импортируемых функций – порядок описания, допустимые параметры, порядок поиска и соглашения о связях импортируемых функций;
- Ошибки выполнения – получение информации об ошибках выполнения и критические ошибки.

Эксперты, пользовательские индикаторы и скрипты прикрепляются к одному из открытых графиков путем перетаскивания мышью из окна "Навигатор" клиентского терминала на соответствующий график (технология Drag'n'Drop). MQL5-программы могут работать только при включенном клиентском терминале.

Для того чтобы эксперт прекратил свою работу, его необходимо удалить с графика. Для этого из контекстного меню графика следует выбрать "Список экспертов", далее выбрать сответник из списка и нажать кнопку "Удалить". На работу советника также влияет состояние кнопки "Авто-торговля".

Для того чтобы пользовательский индикатор прекратил работу, его необходимо удалить с графика.

Пользовательские индикаторы и советники работают до тех пор, пока их явно не удалят с графика; информация о прикрепленных советниках и пользовательских индикаторах сохраняется между запусками клиентского терминала.

Скрипты выполняются однократно и удаляются автоматически по завершению своей работы, либо по закрытию или изменению состояния текущего графика, либо по завершению работы клиентского терминала. При повторном запуске клиентского терминала скрипты не запускаются, так как информация о них не сохраняется.

На одном графике могут работать максимум по одному эксперту и скрипту и неограниченное количество индикаторов.

Сервисы для своей работы не требуют привязки к графику и предназначены для выполнения вспомогательных функций. Например, в сервисе можно создать пользовательский символ, открыть график созданного символа, и затем в бесконечном цикле получать для него данные при помощи сетевых функций и непрерывно обновлять.

© 2000-2025, MetaQuotes Ltd.

Выполнение программ

Каждый скрипт, сервис и эксперт работает в собственном отдельном потоке. Все индикаторы, рассчитываемые на одном символе, даже если они запущены на разных графиках, работают в одном потоке. Таким образом, все индикаторы на одном символе делят между собой ресурсы одного потока.

В одном потоке с индикаторами также последовательно выполняются остальные действия по данному символу - обработка тиков и синхронизация истории. Это означает, что если в индикаторе выполняется бесконечное действие, все остальные события по его символу никогда не выполнятся.

При запуске эксперта ему необходимо обеспечить актуальное торговое окружение, доступность истории по данному символу и периоду, а также произвести синхронизацию между терминалом и сервером. На эти процедуры терминал предоставляет эксперту отсрочку запуска не более чем в 5 секунд, по истечении которых эксперт будет запущен с теми данными, что удалось подготовить. Поэтому при отсутствии связи с сервером это может привести к задержке запуска эксперта.

Краткая сводка по программам на MQL5 приведена в таблице:

Программа	Выполнение	Примечание
Сервис	В собственном потоке, сколько сервисов - столько потоков выполнения для них	Зацикленный сервис не может нарушить работу других программ
Скрипт	В собственном потоке, сколько скриптов - столько потоков выполнения для них	Зацикленный скрипт не может нарушить работу других программ
Эксперт	В собственном потоке, сколько экспертов - столько потоков выполнения для них	Зацикленный эксперт не может нарушить работу других программ
Индикатор	Один поток выполнения для всех индикаторов на одном символе. Сколько символов с индикаторами - столько потоков выполнения для них	Бесконечный цикл в одном индикаторе остановит работу всех остальных индикаторов на этом символе

Сразу после присоединения программы к графику производится ее загрузка в память клиентского терминала и инициализация глобальных переменных. Если какая-либо глобальная переменная типа класса имеет конструктор, то этот конструктор будет вызван в процессе инициализации глобальных переменных.

После этого программа находится в состоянии ожидания события от клиентского терминала. У каждой mql5-программы должна быть хотя бы одна функция-обработчик события, в противном случае загруженная программа выполняться не будет. Функции-обработчики событий имеют предопределенные имена, предопределенные наборы параметров и предопределенные типы возврата.

© 2000-2025, MetaQuotes Ltd.

Тип	Имя функции	Параметры	Применение	Примечание
int	OnInit	нет	эксперты и индикаторы	Обработчик события Init. Допускается тип возвращаемого значения void.
void	OnDeinit	const int reason	эксперты и индикаторы	Обработчик события Deinit.
void	OnStart	нет	скрипты и сервисы	Обработчик события 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[]	индикаторы	Обработчик события Calculate на всех ценовых данных.
int	OnCalculate	const int rates_total, const int prev_calculated, const int begin, const double &price[]	индикаторы	Обработчик события Calculate на одном массиве данных. В индикаторе не может одновременно присутствовать 2 обработчика Calculate. В этом случае будет работать только обработчик события Calculate на одном массиве данных.
void	OnTick	нет	эксперты	Обработчик события NewTick. Пока идет обработка события прихода нового тика другие события этого типа не приходят.
void	OnTimer	нет	эксперты и	Обработчик события Timer.

© 2000-2025, MetaQuotes Ltd.

Тип	Имя функции	Параметры	Применение	Примечание
			индикаторы	
void	OnTrade	нет	эксперты	Обработчик события Trade.
double	OnTester	нет	эксперты	Обработчики события Tester
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	эксперты и индикаторы	Обработчик события ChartEvent.
void	OnBookEvent	const string &symbol_name	эксперты и индикаторы	Обработчик события BookEvent.

Клиентский терминал отсылает возникающие события в соответствующие открытые графики. Также события могут генерироваться графиками (события графика) либо mql5-программами (пользовательские события). Генерацию событий создания и удаления графических объектов на графике можно включать и отключать заданием свойств графика CHART_EVENT_OBJECT_CREATE и CHART_EVENT_OBJECT_DELETE. Каждая mql5-программа и каждый график имеют свою собственную очередь событий, куда складываются все вновь поступающие события.

Программа получает события только от графика, на котором она запущена. Все события обрабатываются одно за другим в порядке поступления. Если в очереди уже есть событие NewTick либо это событие находится в состоянии обработки, то новое событие NewTick в очередь mql5-программы не ставится. Аналогично, если в очереди mql5-программы уже находится событие ChartEvent или такое событие обрабатывается, то новое событие такого типа не ставится в очередь. Обработка событий таймера производится по такой же схеме – если в очереди находится или уже обрабатывается событие Timer, то новое событие таймера не ставится в очередь.

Очереди событий имеют ограниченный, но достаточный размер, поэтому переполнение очереди для корректно написанной программы маловероятно. При переполнении очереди новые события отбрасываются без постановки в очередь.

Крайне не рекомендуется использовать бесконечные циклы для обработки событий. Исключением из этого правила могут быть только скрипты и сервисы, которые обрабатывают одно единственное событие Start.

Библиотеки не обрабатывают никаких событий.

Запрет на использование функций в индикаторах и экспертах

© 2000-2025, MetaQuotes Ltd.

Индикаторы, скрипты и эксперты являются исполняемыми программами на MQL5 и предназначены для различных типов задач. Поэтому существует ограничение на использование определенных функций в зависимости от типа программы. В индикаторах запрещены следующие функции:

- OrderCalcMargin();
- OrderCalcProfit();
- OrderCheck();
- OrderSend();
- SendFTP();
- Sleep();
- ExpertRemove();
- MessageBox().

В свою очередь, в экспертах и скриптах запрещены все функции, предназначенные для индикаторов:

- SetIndexBuffer();
- IndicatorSetDouble();
- IndicatorSetInteger();
- IndicatorSetString();
- PlotIndexSetDouble();
- PlotIndexSetInteger();
- PlotIndexSetString();
- PlotIndexGetInteger.

Библиотека не является самостоятельной программой и выполняется в контексте вызвавшей её MQL5-программы: скрипт, индикатор или эксперт. Соответственно, на вызванную библиотеку распространяются указанные выше ограничения.

Запрет на использование функций в сервисах

Сервисы не принимают никаких событий, так как не имеют привязки к графику. В сервисах запрещены следующие функции:

ExpertRemove();

EventSetMillisecondTimer();

EventSetTimer();

EventKillTimer();

SetIndexBuffer();

IndicatorSetDouble();

IndicatorSetInteger();

IndicatorSetString();

PlotIndexSetDouble();

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetInteger();

PlotIndexSetString();

PlotIndexGetInteger();

Загрузка и выгрузка индикаторов

Индикаторы загружаются в следующих случаях:

- прикрепление индикатора к графику;
- запуск терминала (если индикатор был прикреплен к графику перед предыдущим закрытием терминала);
- загрузка шаблона (если в шаблоне указан прикрепленный к графику индикатор);
- смена профиля (если индикатор прикреплен к одному из графиков профиля);
- смена символа и/или периода графика, к которому прикреплен индикатор;
- смена счета, к которому подключен терминал;
- после удачной перекомпиляции индикатора, если данный индикатор был прикреплен к графику.
- изменение входных параметров индикатора.

Индикаторы выгружаются в следующих случаях:

- при откреплении индикатора от графика;
- закрытие терминала (если индикатор был прикреплен к графику);
- загрузка шаблона, если к графику прикреплен индикатор;
- закрытие графика, к которому был прикреплен индикатор;
- смена профиля, если индикатор прикреплен к одному из графиков сменяемого профиля;
- смена символа и/или периода графика, к которому прикреплен индикатор;
- смена счета, к которому подключен терминал;
- изменение входных параметров индикатора.

Загрузка и выгрузка экспертов

Загрузка эксперта производится в следующих случаях:

- прикрепление эксперта к графику;
- запуск терминала (если эксперт был прикреплен к графику перед предыдущим закрытием терминала);
- загрузка шаблона (если в шаблоне указан прикрепленный к графику эксперт);
- после удачной перекомпиляции эксперта, если данный эксперт был прикреплен к графику.
- смена профиля (если эксперт прикреплен к одному из графиков профиля);
- подключение к счету, даже если номер счета не менялся (если эксперт был прикреплен к графику перед авторизацией терминала на сервере).

Выгрузка эксперта, прикрепленного к графику, производится в следующих случаях:

© 2000-2025, MetaQuotes Ltd.

- при откреплении эксперта от графика;
- при прикреплении эксперта к графику – если на данном графике был уже другой эксперт, то этот эксперт выгружается;
- закрытие терминала (если эксперт был прикреплен к графику);
- загрузка шаблона, если к графику прикреплен эксперт;
- закрытие графика, к которому был прикреплен эксперт;
- смена профиля, если эксперт прикреплен к одному из графиков сменяемого профиля;
- смена счета, к которому подключен терминал (если эксперт был прикреплен к графику перед авторизацией терминала на сервере);
- вызов функции ExpertRemove().

При смене символа или таймфрейма графика, к которому эксперт прикреплен, выгрузка и загрузка эксперта не производится. При этом последовательно вызываются обработчики OnDeinit() на старом символе/таймфрейме и OnInit() на новом символе/таймфрейме (если они есть), значения глобальных переменных и статических переменных не сбрасываются. Все события, поступившие для эксперта до завершения инициализации (функции OnInit()), пропускаются.

Загрузка и выгрузка скриптов

Скрипты загружаются сразу после прикрепления к графику и выгружаются сразу после окончания своей работы. При этом функции OnInit() и OnDeinit() для скриптов не вызываются.

При выгрузке программы (удалении программы с графика) происходит деинициализация глобальных переменных и уничтожение очереди сообщений. В этом случае деинициализация означает освобождение переменных типа string, освобождение объектов динамических массивов и вызов деструкторов при их наличии.

Загрузка и выгрузка сервисов

Сервисы загружаются сразу после запуска терминала, если в момент остановки терминала они были запущены. Сервисы выгружаются сразу после окончания своей работы.

Сервисы имеют единственный обработчик OnStart(), в котором вы можете организовать бесконечный цикл получения и обработки данных, например - создание и обновление пользовательских символов при помощи сетевых функций.

В отличие от советников, индикаторов и скриптов, сервисы не привязаны к конкретному графику - поэтому для запуска сервиса предусмотрен отдельный механизм. Создание нового экземпляра сервиса производится из Навигатора с помощью команды "Добавить сервис". Для запуска, остановки и удаления экземпляра сервиса используйте его меню. Для управления всеми экземплярами, используйте меню самого сервиса.

© 2000-2025, MetaQuotes Ltd.

Для лучшего понимания работы эксперта рекомендуется скомпилировать код приведенного в примере эксперта и произвести действия по загрузке/выгрузке экспертов, смене шаблона, символа, таймфрейма и так далее:

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 constructor"); }
   ~CTestClass() { Print("CTestClass destructor"); }
  };
CTestClass global;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Initialization");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print("Deinitialization with reason ",reason);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

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

© 2000-2025, MetaQuotes Ltd.

Скрипты загружаются сразу после прикрепления к графику и выгружаются сразу после окончания своей работы.

Смотри также

События клиентского терминала, Функции обработки событий

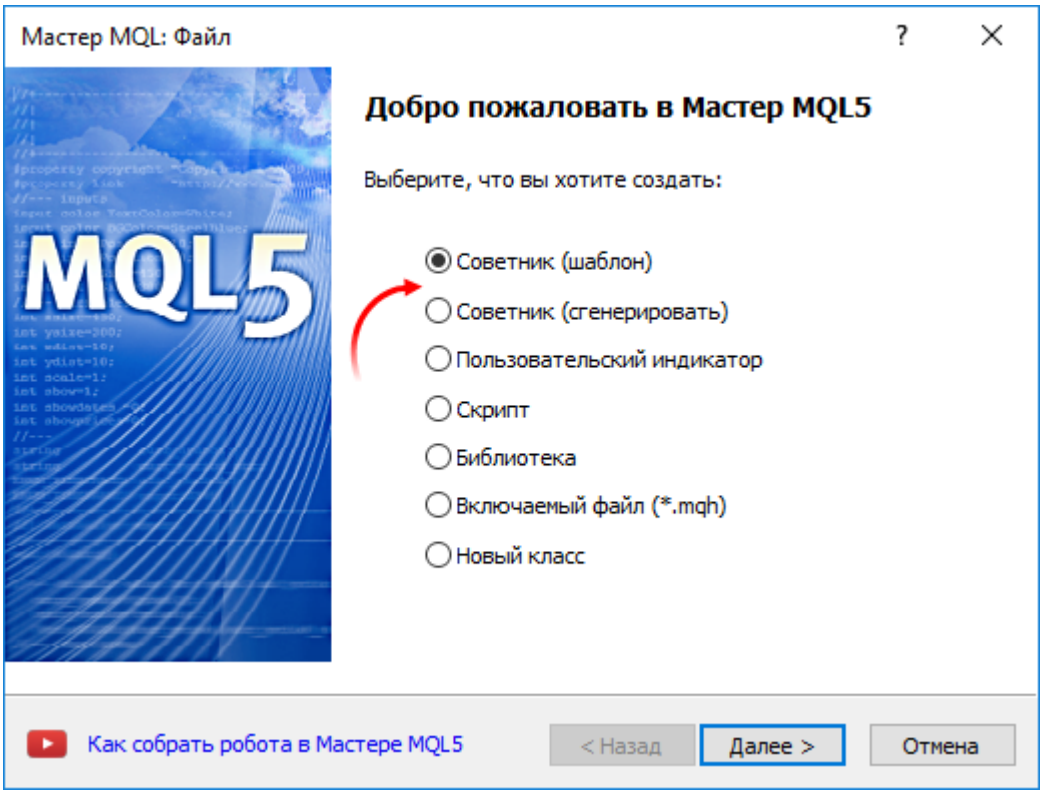
© 2000-2025, MetaQuotes Ltd.

Разрешение на торговлю

Автоматизация торговли

В языке MQL5 существует специальная группа торговых функций, с помощью которых можно создавать автоматизированные торговые системы. Программы для автоматической торговли без участия человека называются экспертами (Expert Advisor) или торговыми роботами. Для создания эксперта в редакторе MetaEditor запустите мастер создания советников MQL5 Wizard и выберите один из двух вариантов:

- Expert Advisor (template) - позволяет создать шаблон с готовыми функциями обработки событий, который необходимо дополнить всем необходимым функционалом с помощью самостоятельного программирования.
- Expert Advisor (generate) - дает возможность создать полностью готового для работы торгового робота, просто выбирая необходимые вам модули: модуль формирования торговых сигналов, модуль управления капиталом и модуль подтягивания защитного стопа для открытых позиций (Trailing Stop).



Торговые функции могут работать только в экспертах и скриптах, для индикаторов торговля запрещена.

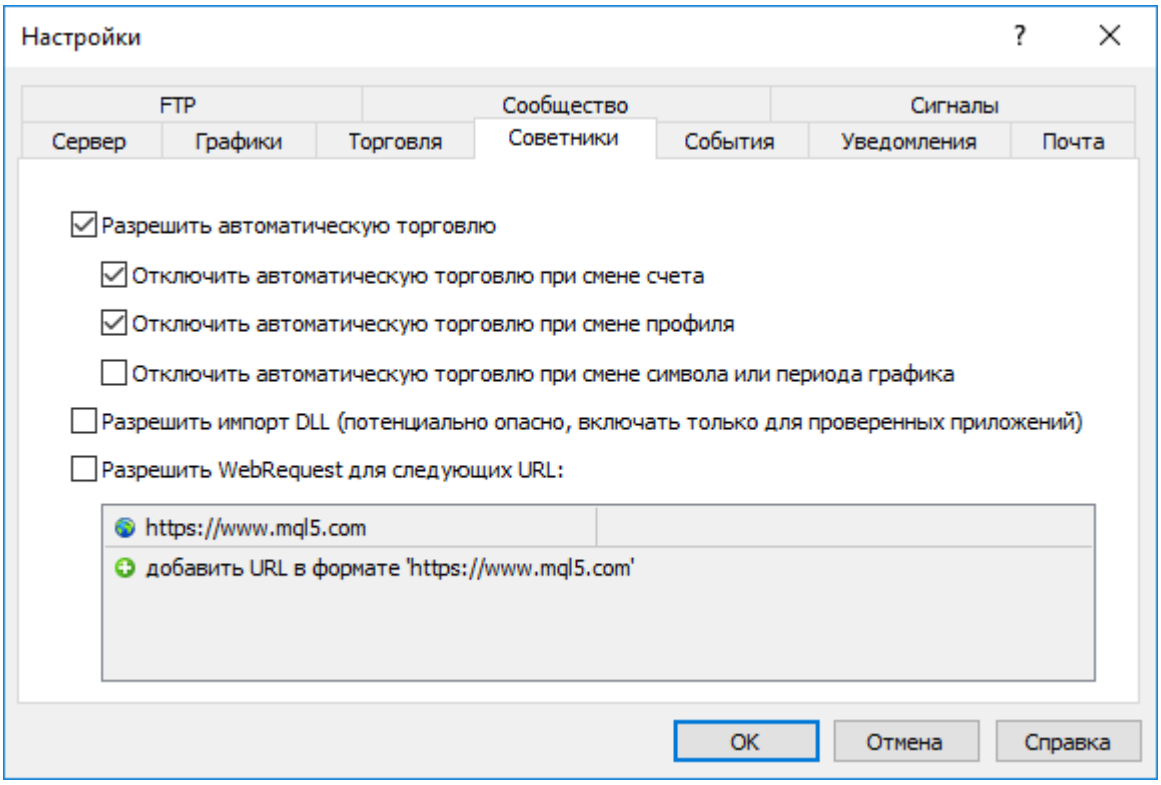
Проверка разрешения на автоматическую торговлю

Для создания надежного эксперта, который мог бы работать в автоматическом режиме без контроля человека, необходимо организовать множество необходимых проверок. В первую очередь необходимо программным путем проверять, есть ли вообще разрешение на совершение торговых операций. Проверка разрешения на торговлю является базовой и обязательной при разработке любой автоматической системы.

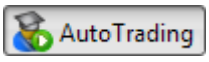
© 2000-2025, MetaQuotes Ltd.

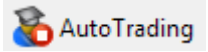
Проверка на разрешение автоматической торговли в самом терминале

В настройках терминала можно запретить или разрешить автоматическую торговлю для всех программ.



Переключать опцию разрешения автоматической торговли также можно прямо из Стандартной панели терминала:

- 

- автоматическая торговля разрешена, торговые функции в запущенных программах разрешены для использования.
- 

- автоматическая торговля запрещена, при этом сами запущенные программы будут работать, хотя торговые функции не смогут выполняться.

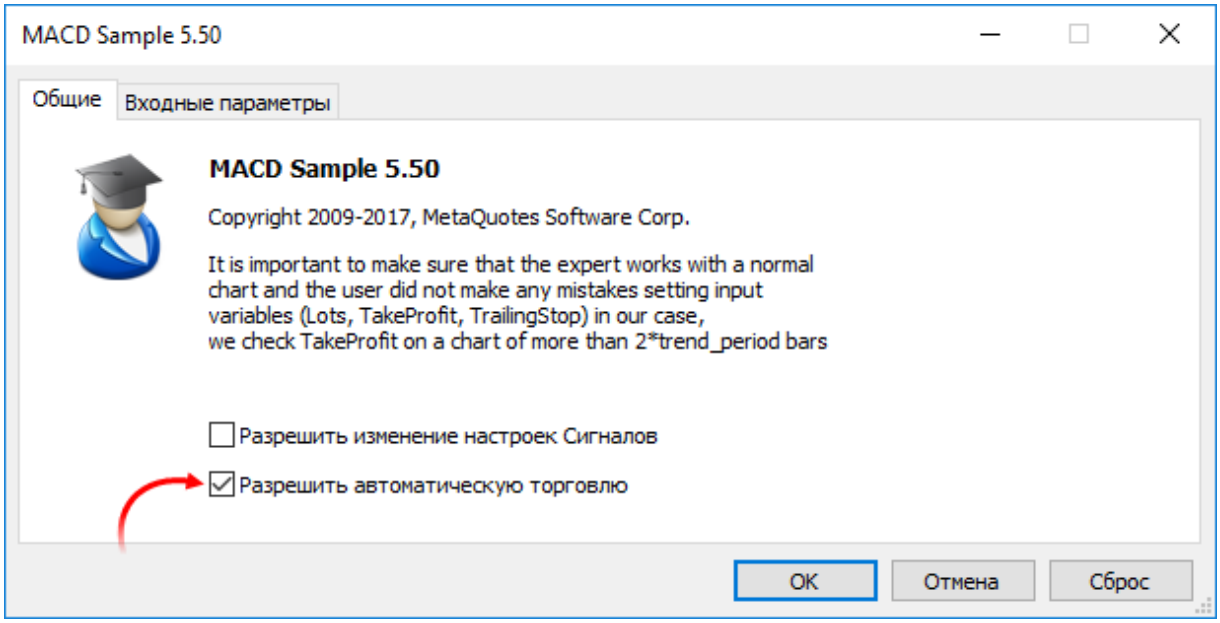
Пример проверки:

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Проверьте в настройках терминала разрешение на автоматическую торговлю!");
```

Проверка разрешения на торговлю для данного запущенного эксперта/скрипта

При запуске программы можно разрешить или запретить автоматическую торговлю конкретно для нее. Для этого есть отдельная специальная настройка в свойствах программы.

© 2000-2025, MetaQuotes Ltd.



Пример проверки:

```
if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Проверьте в настройках терминала разрешение на автоматическую торговлю!")
else
  {
   if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
      Alert("Автоматическая торговля запрещена в свойствах программы для ",__FILE__
  }
```

Проверка разрешения на торговлю любым экспертам/скриптам для данного счета

Запрет на автоматическую торговлю может быть установлен на стороне торгового сервера. Пример проверки такой ситуации:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
   Alert("Автоматическая торговля запрещена для счета ",AccountInfoInteger(ACCOUNT_
   " на стороне торгового сервера");
```

Если для торгового счета запрещена автоматическая торговля, то торговые операции из экспертов/скриптов выполняться не будут.

Проверка разрешения торговли для данного счета

Возможны случаи, когда для конкретного торгового счета запрещены любые торговые операции - нельзя торговать ни вручную, ни с помощью экспертов. Пример проверки ситуации, когда к торговому счету подключились с помощью инвесторского пароля:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
   Comment("Торговля запрещена для счета ",AccountInfoInteger(ACCOUNT_LOGIN),
         ".\n Возможно, подключение к торговому счету произведено по инвест паролю.
         "\n Проверьте журнал терминала, есть ли там такая запись:",
```

© 2000-2025, MetaQuotes Ltd.

```
            "\n\'",AccountInfoInteger(ACCOUNT_LOGIN),"\': trading has been disabled -
```

AccountInfoInteger(ACCOUNT_TRADE_ALLOWED) может возвращать false в следующих случаях:

- нет соединения с торговым сервером. Можно проверить с помощью TerminalInfoInteger(TERMINAL_CONNECTED);
- торговый счет переведен в режим read-only (отправлен в архив);
- торговля на счете запрещена на стороне торгового сервера;
- подключение к торговому счете произведено в режиме инвестора.

Смотри также

Состояние клиентского терминала, Информация о счете, Информация о запущенной MQL5-программе

© 2000-2025, MetaQuotes Ltd.

События клиентского терминала

Init

Сразу же после того, как клиентский терминал загрузит программу (эксперт или пользовательский индикатор) и запустит процесс инициализации глобальных переменных, будет послано событие Init, которое обрабатывается функцией OnInit(), если она есть. Это событие также генерируется после смены финансового инструмента и/или периода графика, после перекомпиляции программы в редакторе MetaEditor, после смены входных параметров из окна настройки эксперта или пользовательского индикатора. Советник также инициализируется после смены счета. Для скриптов событие Init не генерируется.

Deinit

Перед деинициализацией глобальных переменных и выгрузкой программы (эксперт или пользовательский индикатор) клиентский терминал посылает программе событие Deinit. Событие Deinit также генерируется при завершении работы клиентского терминала, при закрытии графика, непосредственно перед сменой финансового инструмента и/или периода графика, при удачной перекомпиляции программы, при смене входных параметров, а также при смене счета.

Причину деинициализации можно получить из параметра, переданного в функцию OnDeinit(). Выполнение функции OnDeinit() ограничивается 2.5 секундами. Если за это время функция не закончила свою работу, то ее выполнение завершается принудительно. Для скриптов событие Deinit не генерируется.

Start

Событие Start – это специальное событие для запуска скрипта или сервиса после его загрузки. Это событие обрабатывается функцией OnStart. Событие Start экспертам и пользовательским индикаторам не посылается.

NewTick

Событие NewTick генерируется при поступлении новых котировок и обрабатывается функцией OnTick() у присоединенных советников. Если при поступлении новой котировки выполнялась функция OnTick, запущенная на предыдущей котировке, то пришедшая котировка будет проигнорирована советником, так как соответствующее событие не будет поставлено в очередь событий эксперта.

Все пришедшие во время выполнения программы новые котировки программой игнорируются до тех пор, пока не завершится очередное выполнение функции OnTick(). После этого функция будет запущена только после прихода очередной новой котировки.

Событие NewTick генерируется независимо от того, запрещена или разрешена автоматическая торговля (кнопка "Разрешить/запретить Авто-торговлю"). Запрет автоматической торговли означает только запрет на отправку торговых запросов из эксперта, работа эксперта не прекращается.

Запрет автоматической торговли путем нажатия на указанную кнопку не прерывает текущее выполнение функции OnTick().

Calculate

© 2000-2025, MetaQuotes Ltd.

Событие Calculate генерируется только для индикаторов сразу после посылки события Init и при любом изменении ценовых данных. Обрабатывается функцией OnCalculate.

Timer

Событие Timer периодически генерируется клиентским терминалом для эксперта, который активизировал таймер при помощи функции EventSetTimer. Обычно эта функция вызывается в функции OnInit. Обработка события Timer производится функцией OnTimer. При завершении работы эксперта необходимо уничтожить созданный таймер при помощи EventKillTimer, которую обычно вызывают в функции OnDeinit.

Trade

Событие Trade генерируется при завершении торговой операции на торговом сервере. Обработка события Trade производится функцией OnTrade() для следующих торговых операций:

- установка, модификация или удаление отложенного ордера;
- отмена отложенного ордера при нехватке средств либо по истечении срока действия;
- срабатывание отложенного ордера;
- открытие, добавление или закрытие позиции (или части позиции);
- модификация открытой позиции (изменение стопов).

TradeTransaction

В результате выполнения определенных действий с торговым счетом, его состояние изменяется. К таким действиям относятся:

- Отсылка торгового запроса любым MQL5-приложением в клиентском терминале при помощи функций OrderSend и OrderSendAsync и его последующее исполнение;
- Отсылка торгового запроса через графический интерфейс терминала и его последующее исполнение;
- Срабатывание отложенных ордеров и стоп-ордеров на сервере;
- Выполнение операций на стороне торгового сервера.

В результате данных действий, для счета выполняются торговые транзакции:

- обработка торгового запроса;
- изменение открытых ордеров;
- изменение истории ордеров;
- изменение истории сделок;
- изменение позиций.

Например, при отсылке рыночного ордера на покупку, он обрабатывается, для счета создается соответствующий ордер на покупку, происходит исполнение ордера, его удаление из списка открытых, добавление в историю ордеров, далее добавляется соответствующая сделка в историю и создается новая позиция. Все эти действия являются торговыми транзакциями. Приход каждой такой транзакции в терминал является событием TradeTransaction. Данное событие обрабатывается функцией OnTradeTransaction.

Tester

© 2000-2025, MetaQuotes Ltd.

Событие Tester генерируется по окончании тестирования эксперта на исторических данных. Обработка события Tester производится функцией OnTester().

TesterInit

Событие TesterInit генерируется при запуске оптимизации в тестере стратегий перед самым первым проходом. Обработка события TesterInit производится функцией OnTesterInit().

TesterPass

Событие TesterPass генерируется при поступлении нового фрейма данных. Обработка события TesterPass производится функцией OnTesterPass().

TesterDeinit

Событие TesterDeinit генерируется по окончании оптимизации эксперта в тестере стратегий. Обработка события TesterDeinit производится функцией OnTesterDeinit().

ChartEvent

Событие ChartEvent генерируется клиентским терминалом при работе пользователя с графиком:

- нажатия клавиатуры, когда окно графика находится в фокусе;
- создание графического объекта;
- удаление графического объекта;
- щелчок мыши на графическом объекте, принадлежащего графику;
- перемещение графического объекта при помощи мыши;
- окончание редактирования текста в поле ввода графического объекта LabelEdit.

Также существует пользовательское событие ChartEvent, которое может послать эксперту любая mql5-программа при помощи функции EventChartCustom. Событие обрабатывается функцией OnChartEvent.

BookEvent

Событие BookEvent генерируется клиентским терминалом при изменении состояния стакана цен и обрабатывается функцией OnBookEvent. Для того чтобы клиентский терминал начал генерировать события BookEvent по конкретному символу, достаточно предварительно подписаться на получение этих событий для этого символа с помощью функции MarketBookAdd.

Для того чтобы отписаться от получения события BookEvent по символу, необходимо вызывать функцию MarketBookRelease. Событие BookEvent является широковещательным – это означает, что достаточно одному эксперту подписаться на получение события BookEvent с помощью функции MarketBookAdd, и все остальные эксперты, имеющие обработчик OnBookEvent, будут получать это событие. Поэтому необходимо анализировать имя символа, которое передается в обработчик в качестве параметра.

Смотри также

Функции обработки событий, Выполнение программ

© 2000-2025, MetaQuotes Ltd.

Ресурсы

Использование графики и звука в программах на MQL5

Программы на MQL5 позволяют работать со звуковыми и графическими файлами:

- PlaySound() воспроизводит звуковой файл;
- ObjectCreate() позволяет с помощью графических объектов OBJ_BITMAP и OBJ_BITMAP_LABEL создавать пользовательские интерфейсы.

PlaySound()

Пример вызова функции PlaySound():

```
//+------------------------------------------------------------------+
//| функция вызывает штатную OrderSend() и воспроизводит звук        |
//+------------------------------------------------------------------+
void OrderSendWithAudio(MqlTradeRequest  &request, MqlTradeResult &result)
  {
  //--- отправим запрос на сервер
   OrderSend(request,result);
   //--- если запрос принят, играем звук Ok.wav
   if(result.retcode==TRADE_RETCODE_PLACED)
      PlaySound("Ok.wav");
   //--- при неудаче выдаем тревожный звук из файла timeout.wav
   else
      PlaySound("timeout.wav");
  }
```

В данном примере показано как проигрывать звуки из файлов Ok.wav и timeoit.wav, входящих в стандартную поставку терминала. Эти файлы находятся в папке **каталог_терминала\Sounds**. Здесь **каталог_терминала** означает папку, из которой запущен клиентский терминал MetaTrader 5. Программным путем из mql5-программы каталог терминала можно узнать следующим образом:

```
//--- Папка, в которой хранятся данные терминала
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

Можно использовать звуковые файлы не только из папки **каталог_терминала\Sounds**, но и из любой подпапки, находящейся в папке **каталог_данных_терминала\MQL5**. Расположение каталога данных терминала на компьютере можно выяснить через меню терминала "Файл"-"Открыть каталог данных" или программным путем:

```
//--- Папка, в которой хранятся данные терминала
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

Например, если звуковой файл Demo.wav лежит в папке каталог_данных_терминала\MQL5\Files, то вызов PlaySound() должен быть записан таким образом:

```
//--- проиграем звуковой файл Demo.wav из папки каталог_данных_терминала\MQL5\Files\
   PlaySound("\\Files\\Demo.wav");
```

© 2000-2025, MetaQuotes Ltd.

Обратите внимание на то, что в комментарии путь к файлу написан с использованием символа "\", а в самой функции для разделения папок в пути используется последовательность "\\".

При указании пути всегда используйте только двойную обратную косую черту в качестве разделителя, так как одиночная обратная черта является управляющим символом для компилятора при разборе константных строк и символьных констант в исходном коде программы.

Для остановки воспроизведения файла нужно вызвать функцию PlaySound() с параметром NULL:

```
//--- вызов PlaySound() с параметром NULL останавливает воспроизведение звука
   PlaySound(NULL);
```

ObjectCreate()

Пример эксперта, который с помощью функции ObjectCreate() создает объект "Графическая метка" (OBJ_BITMAP_LABEL).

```
string label_name="currency_label";         // имя объекта OBJ_BITMAP_LABEL
string euro      ="\\Images\\euro.bmp";     // путь к файлу каталог_данных_терминала\MQ
string dollar    ="\\Images\\dollar.bmp";   // путь к файлу каталог_данных_терминала\MQ
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- создадим кнопку OBJ_BITMAP_LABEL, если ее еще нет
   if(ObjectFind(0,label_name)<0)
     {
      //--- попробуем создать объект OBJ_BITMAP_LABEL
      bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
      if(created)
        {
         //--- привяжем кнопку к правому верхнему углу графика
         ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
         //--- теперь настроим свойства объекта
         ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
         ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
         //--- сбросим код последней ошибки в 0
         ResetLastError();
         //--- загрузим картинку для состояния кнопки "Нажата"
         bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
         //--- проверим результат
         if(!set)
           {
            PrintFormat("Не удалось загрузить картинку из файла %s. Код ошибки %d",eur
           }
         ResetLastError();
         //--- загрузим картинку для состояния кнопки "Отжата"
```

© 2000-2025, MetaQuotes Ltd.

```
         set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

         if(!set)
           {
            PrintFormat("Не удалось загрузить картинку из файла %s. Код ошибки %d",do
           }
         //--- отдадим графику команду на обновление, чтобы кнопка появилась сразу же,
         ChartRedraw(0);
        }
      else
        {
         //--- объект создать не удалось, сообщим об этом
         PrintFormat("Не удалось создать объект OBJ_BITMAP_LABEL. Код ошибки %d",GetLa
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- удалим объект с графика
   ObjectDelete(0,label_name);
  }
```

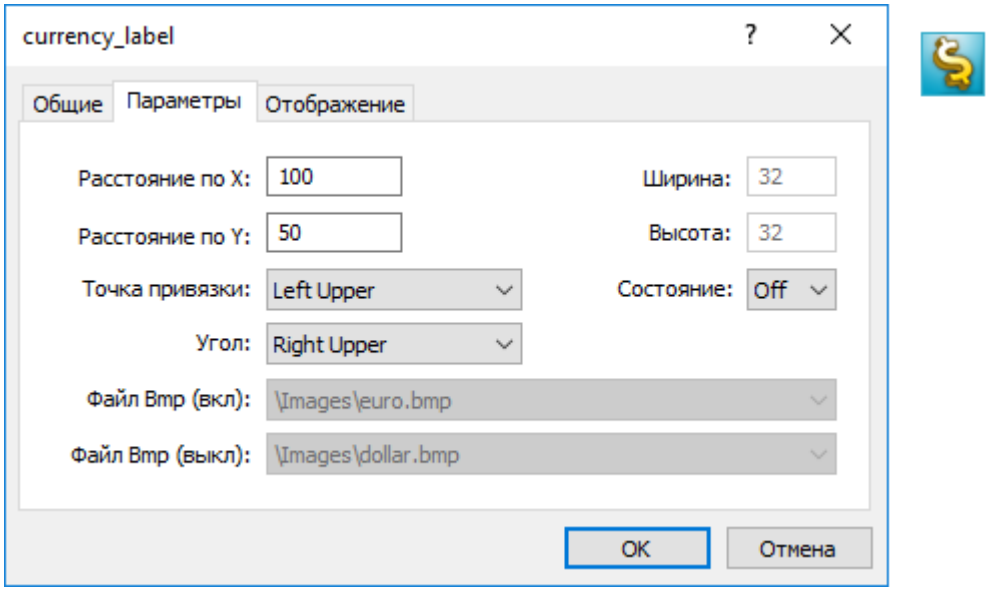
Создание и настройка графического объекта с именем currency_label происходят в функции OnInit(). Пути к загружаемым графическим файлам заданы в глобальных переменных **euro** и **dollar**, в качестве разделителя использована двойная обратная косая черта:

```
string euro      ="\\Images\\euro.bmp";     // путь к файлу каталог_данных_терминала\MQ
string dollar    ="\\Images\\dollar.bmp";   // путь к файлу каталог_данных_терминала\MQ
```

Сами файлы при этом находятся в папке **каталог_данных_терминала**\MQL5\Images.

Объект OBJ_BITMAP_LABEL фактически представляет собою кнопку, которая в зависимости от состояния (нажата или отжата) может отображать одну из двух картинок: euro.bmp или dollar.bmp.

© 2000-2025, MetaQuotes Ltd.



Размеры кнопки с графическим интерфейсом автоматически устанавливаются под размер отображаемой картинки. Смена изображения производится при нажатии левой кнопкой мышки на объекте OBJ_BITMAP_LABEL (**в свойствах должна быть выбрана опция "Отключить выделение"**). Объект OBJ_BITMAP создается аналогичным образом и предназначен для создания фона с требуемым рисунком.

Значение свойства OBJPROP_BMPFILE, которое отвечает за вид объектов OBJ_BITMAP и OBJ_BITMAP_LABEL, можно менять динамически. Это позволяет создавать разнообразные интерактивные пользовательские интерфейсы для mql5-программ.

Включение ресурсов в исполняемые файлы при компиляции mql5-программ

Для работы mql5-программы может потребоваться множество разнообразных загружаемых ресурсов в виде файлов изображений и звуков. Для того чтобы исключить необходимость переноса всех этих файлов при передаче исполняемой программы на MQL5, следует использовать директиву компилятора #resource:

```
#resource путь_к_файлу_ресурса
```

Команда #resource указывает компилятору, что ресурс по указанному пути **путь_к_файлу_ресурса** нужно включить в исполняемый файл EX5. Таким образом, все необходимые картинки и звуки можно поместить непосредственно в EX5-файл и для запуска программы в другом терминале не потребуется передавать все используемые в ней отдельные файлы. Любой EX5-файл может содержать ресурсы, и любая EX5-программа может использовать ресурсы из другой EX5-программы.

Файлы в формате BMP и WAV перед включением в исполняемый EX5 файл автоматически сжимаются. Это означает, что использование ресурсов не только позволяет создавать полноценные программы на MQL5, но и уменьшает общий размер требуемых терминалу файлов при использовании графики и звука по сравнению с обычным способом написания mql5-программ.

Размер файла ресурса не может быть больше 128 Mb.

© 2000-2025, MetaQuotes Ltd.

Поиск указанных ресурсов компилятором

Ресурс вставляется командой #resource "<путь к файлу ресурса>"

```
#resource "<путь_к_файлу_ресурса>"
```

Длина константной строки <путь_к_файлу_ресурса> не должна превышать 63 символа.

Компилятор ищет ресурс по указанному пути в следующей последовательности:

- если в начале пути стоит разделитель обратная косая черта "\" (пишется "\\"), то ресурс ищется относительно каталога **каталог_данных_терминала**\MQL5\,
- если обратной косой черты нет, то ресурс ищется относительно расположения исходного файла, в котором этот ресурс прописывается.

В пути ресурса недопустимо использовать подстроки "..\\" и ":\\".

Примеры включения ресурсов:

```
//--- правильное указание ресурсов
#resource "\\Images\\euro.bmp" // euro.bmp находится в каталог_данных_терминала\MQL5\I
#resource "picture.bmp"        // picture.bmp находится в том же каталоге, где и исход
#resource "Resource\\map.bmp"  // ресурс находится в папке каталог_исходного_файла\Res

//--- неправильное указание ресурсов
#resource ":picture_2.bmp"     // нельзя использовать ":"
#resource "..\\picture_3.bmp"  // нельзя использовать ".."
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" //больш
```

Использование ресурсов

Имя ресурса

После того как ресурс объявлен директивой #resource, его можно использовать в любой части программы. Именем ресурса становится его путь без косой черты в начале строки, задающей путь к ресурсу. Для использования своего ресурса в коде нужно перед именем ресурса добавлять специальный признак "::".

Примеры:

```
//--- примеры указания ресурсов и их имена в комментарии
#resource "\\Images\\euro.bmp"           // имя ресурса - Images\euro.bmp
#resource "picture.bmp"                  // имя ресурса - picture.bmp
#resource "Resource\\map.bmp"            // имя ресурса - Resource\map.bmp
#resource "\\Files\\Pictures\\good.bmp" // имя ресурса - Files\Pictures\good.bmp
#resource "\\Files\\Demo.wav";           // имя ресурса - Files\Demo.wav"
#resource "\\Sounds\\thrill.wav";        // имя ресурса - Sounds\thrill.wav"
...
```

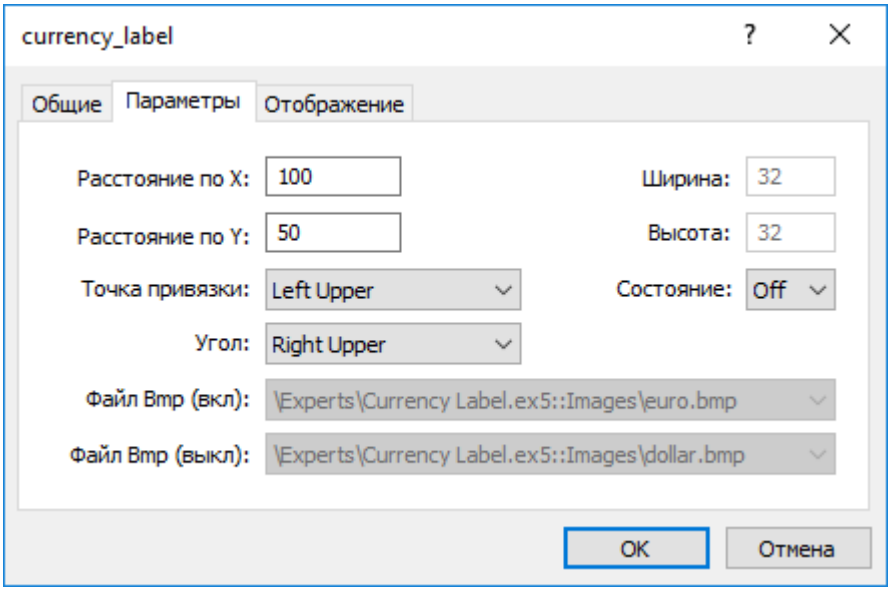
© 2000-2025, MetaQuotes Ltd.

```
//--- использование ресурсов
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");
```

Необходимо отметить, что при установке объектам OBJ_BITMAP и OBJ_BITMAP_LABEL изображения из ресурса, значение свойства OBJPROP_BMPFILE уже нельзя менять вручную. Например, пусть мы используем для создания OBJ_BITMAP_LABEL ресурсы файлов euro.bmp и dollar.bmp.

```
#resource "\\Images\\euro.bmp";    // euro.bmp находится в каталог_данных_терминала\MQ
#resource "\\Images\\dollar.bmp";  // dollar.bmp находится в каталог_данных_терминала\
```

Тогда при просмотре свойств этого объекта мы увидим, что свойства BitMap File (On) и BitMap File (Off) имеют серый цвет и недоступны для изменения вручную:



Использование ресурсов других mql5-программ

Использование ресурсов имеет и другое преимущество - в любой mql5-программе можно использовать ресурсы из любого файла EX5. Таким образом, ресурсы из одного файла EX5 можно использовать во многих других mql5-программах.

Для того чтобы использовать имя ресурса из стороннего файла, его нужно указать в виде <путь_имя_файла_EX5>::<имя_ресурса>. Например, пусть в скрипте Draw_Triangles_Script.mq5 указан ресурс на картинку в файле triangle.bmp:

```
#resource "\\Files\\triangle.bmp"
```

© 2000-2025, MetaQuotes Ltd.

Тогда его имя для использования в самом скрипте будет выглядеть как "Files\triangle.bmp", а для использования к имени ресурса нужно добавить специальный признак "::".

```
//--- использование ресурса в самом скрипте
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

Чтобы использовать этот же ресурс из другой программы, например, эксперта, нужно к имени ресурса дополнительно добавить путь к EX5-файлу относительно папки **каталог_данных_терминала\MQL5** и имя EX5-файла этого скрипта - **Draw_Triangles_Script.ex5**. Пусть скрипт лежит в стандартной папке **каталог_данных_терминала\MQL5**Scripts\, тогда вызов нужно написать таким образом:

```
//--- использование ресурса скрипта в эксперте
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.e
```

Если при обращении к ресурсу в другом EX5-файле не указать путь к этому исполняемому файлу, то исполняемый файл ищется в той же папке, где находится и обратившаяся за ресурсом программа. Это означает, что если в советнике запрашивается ресурс из файла Draw_Triangles_Script.ex5 без указания пути, например, так:

```
//--- запрос ресурса скрипта в эксперте без указания пути
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\\
```

то файл будет искаться в папке **каталог_данных_терминала\MQL5**Experts\, если сам советник находится в папке **каталог_данных_терминала\MQL5**Experts\.

Работа с пользовательскими индикаторами, подключенными в качестве ресурсов

Для работы mql5-программ может потребоваться один или несколько пользовательских индикаторов, все они могут быть включены в код исполняемой mql5-программы. Включение индикаторов в качестве ресурсов позволяет упростить распространение программ.

Пример подключения и использования пользовательского индикатора SampleIndicator.ex5, расположенного в папке: **каталог_данных_терминала**\MQL5\Indicators\:

```
//+------------------------------------------------------------------+
//|                                                     SampleEA.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
```

© 2000-2025, MetaQuotes Ltd.

```
   if(handle_ind==INVALID_HANDLE)
     {
      Print("Expert: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
//---
   return(INIT_SUCCEEDED);
  }
```

Случай, когда пользовательский индикатор в функции OnInit() создает одну или несколько копий себя, требует отдельного рассмотрения. Напомним, что для использования ресурса из mql5-программы его необходимо указывать в виде: **<путь_имя_файла_EX5>::<имя_ресурса>**.

Например, если индикатор SampleIndicator.ex5 включается в советник SampleEA.ex5 в качестве ресурса, то путь к самому себе, указанный при вызове iCustom() в функции инициализации пользовательского индикатора, будет выглядеть следующим образом: "\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5". При явном указании данного пути пользовательский индикатор SampleIndicator.ex5 будет жестко привязан к советнику SampleEA.ex5 и теряет способность самостоятельной работы.

Путь до самого себя можно получить при помощи функции GetRelativeProgramPath(), пример использования которой приведен ниже:

```
//+------------------------------------------------------------------+
//|                                                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()
  {
//--- неправильный способ указания ссылки на себя
//--- string path="\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5";
//--- правильный способ получения ссылки на себя
  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);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| GetRelativeProgramPath                                           |
//+------------------------------------------------------------------+
string GetRelativeProgramPath()
  {
   int pos2;
//--- получаем абсолютный путь к программе
   string path=MQLInfoString(MQL_PROGRAM_PATH);
//--- находим позицию подстроки "\MQL5\"
   int    pos =StringFind(path,"\\MQL5\\");
//--- подстрока не найдена - ошибка
   if(pos<0)
      return(NULL);
//--- пропускаем каталог "\MQL5"
   pos+=5;
//--- пропускаем лишние '\'
   while(StringGetCharacter(path,pos+1)=='\\')
      pos++;
//--- если это ресурс, возвращаем путь относительно MQL5-каталога
   if(StringFind(path,"::",pos)>=0)
      return(StringSubstr(path,pos));
//--- найдем разделитель для первого подкаталога в MQL5 (например, MQL5\Indicators)
//--- если его нет, то вернем путь относительно MQL5-каталога
   if((pos2=StringFind(path,"\\",pos+1))<0)
      return(StringSubstr(path,pos));
//--- вернем путь относительно подкаталога (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[])
  {
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Ресурсные переменные

Ресурсы можно объявлять с помощью ресурсных переменных и обращаться с ними так, как будто они являются переменной соответствующего типа. Формат объявления:

```
#resource путь_к_файлу_ресурса as тип_ресурсной_переменной имя_ресурсной_переменной
```

© 2000-2025, MetaQuotes Ltd.

Примеры объявлений:

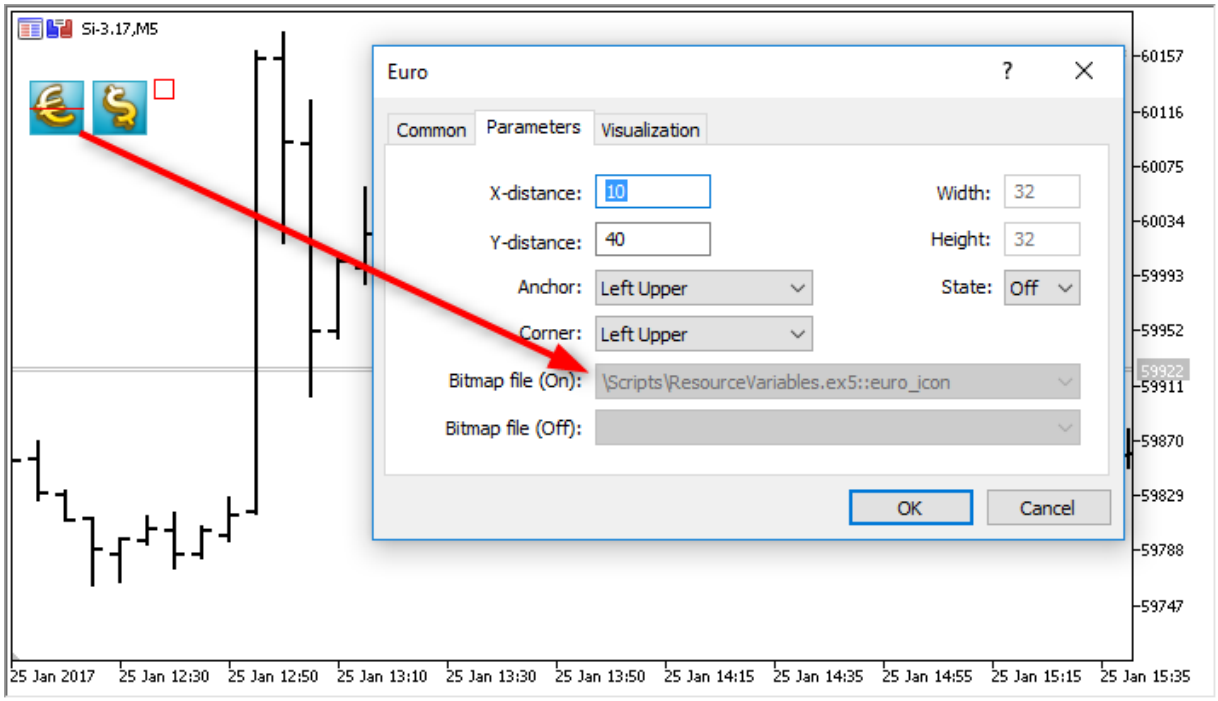
```
#resource "data.bin" as int ExtData[]              // объявление массива числового типа
#resource "data.bin" as MqlRates ExtData[]         // объявление массива простых структ
//--- строки
#resource "data.txt" as string ExtCode             // объявление строки, содержащей дан
//--- графические ресурсы
#resource "image.bmp" as bitmap ExtBitmap[]        // объявление одномерного массива, с
#resource "image.bmp" as bitmap ExtBitmap2[][]     // объявление двумерного массива, со
```

При таком объявлении к данным ресурса можно адресоваться только через переменную, автоматическая адресация через "::<rsource name>" не работает.

```
#resource "\\Images\\euro.bmp" as bitmap euro[][]
#resource "\\Images\\dollar.bmp"
//+------------------------------------------------------------------+
//|  Функция создания объекта OBJ_BITMAP_LABEL с помощью ресурса     |
//+------------------------------------------------------------------+
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);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- выведем размеры картинки [width, height], которая хранится в ресурсной переменно
   Print(ArrayRange(euro,1),", ",ArrayRange(euro,0));
//--- изменим картинку в euro - нарисуем красную горизонтальную полосу посередине
   for(int x=0;x<ArrayRange(euro,1);x++)
      euro[ArrayRange(euro,1)/2][x]=0xFFFF0000;
//--- создадим графический ресурс с помощью ресурсной переменной
   ResourceCreate("euro_icon",euro,ArrayRange(euro,1),ArrayRange(euro,0),0,0,ArrayRang
//--- создадим объект графическая метка Euro, которому выставим картинку из ресурса eu
   Image("Euro","::euro_icon",10,40);
//--- другой способ использования ресурса, рисовать в него мы не можем
   Image("USD","::Images\\dollar.bmp",15+ArrayRange(euro,1),40);
//--- прямой способ адресации к ресурсу euro.bmp недоступен, так как он уже объявлен ч
   Image("E2","::Images\\euro.bmp",20+ArrayRange(euro,1)*2,40); // произойдет ошибка н
  }
```

Результат выполнения скрипта - созданы только два объекта OBJ_BITMAP_LABEL из трех. При этом на изображении первого объекта мы видим красную полоску посредине.

© 2000-2025, MetaQuotes Ltd.



Важным преимуществом использования ресурсов является то, что ресурсные файлы перед включением в исполняемый EX5-файл перед компиляцией автоматически сжимаются. Таким образом, использование ресурсных переменных позволяет не только упаковывать необходимые для работы данные прямо в исполняемый EX5-файл, но и сокращает количество и общий размер файлов по сравнению с обычным способом написания mql5-программ.

Использование ресурсных переменных особенно удобно для публикации продуктов в Маркете.

Особенности

- Специальный тип ресурсной переменной *bitmap* указывает компилятору, что ресурс является графическим изображением. Такие переменные получают тип uint.
- Ресурсная переменная-массив типа *bitmap* может иметь две размерности, в этом случае размер массива будет установлен как [высота_картинки][ширина_картинки]. В случае если указан массив одной размерности, то количество элементов будет равно произведению высота_картинки*ширина_картинки.
- При загрузке 24 битного изображения для всех пикселей изображения компонента альфа-канала устанавливается в значение 255.
- При загрузке 32 битного изображения без альфа-канала, также, для всех пикселей изображения компонента альфаканала устанавливается в значение 255.
- При загрузке 32 битного изображения с альфа-каналом никаких манипуляций с пикселями не происходит.
- Размер файла ресурса не может быть больше 128 Mb.
- Для строковых файлов производится автоматическое определение кодировки по наличию BOM (заголовку). Если BOM отсутствует, то кодировка определяется по содержимому файла. Поддерживаются файлы в кодировке ANSI, UTF-8 и UTF-16. При чтении данных из файлов все строки переводятся в Unicode.

Программы на OpenCL

© 2000-2025, MetaQuotes Ltd.

Использование ресурсных строковых переменных может существенно облегчить написание некоторых программ. Например, вы можете написать код OpenCL-программы в отдельном CL-файле, а затем включить этот файл в виде строки в ресурсы вашей MQL5-программы.

```
#resource "seascape.cl" as string cl_program
...
int context;
if((cl_program=CLProgramCreate(context,cl_program)!=INVALID_HANDLE)
  {
   //--- выполняем дальнейшие действия с OpenCL программой
  }
```

В данном примере без использования ресурсной переменной *cl_program* вам пришлось бы описывать весь код в виде одной большой строковой переменной.

Смотри также

ResourceCreate(), ResourceSave(), PlaySound(), ObjectSetInteger(), ChartApplyTemplate(), Файловые операции

© 2000-2025, MetaQuotes Ltd.

Вызов импортируемых функций

Для импорта функций во время выполнения mql5-программы используется раннее связывание. Это значит, что если в программе есть вызов импортируемой функции, то соответствующий модуль (ex5 или dll) загружается в процессе загрузки программы. Библиотеки MQL5 и DLL выполняются в потоке вызывающего модуля.

Не рекомендуется использовать полностью квалифицированное имя загружаемого модуля вида *Drive:\Directory\FileName.Ext*. Библиотеки MQL5 загружаются из папки *terminal_dir\MQL5\Libraries*. Если библиотека не была найдена, то производится попытка загрузить библиотеку из папки *terminal_dir\experts*.

Системные библиотеки (DLL) загружаются по правилам операционной системы. Если библиотека уже загружена (например, другим экспертом и даже из другого клиентского терминала, запущенного параллельно), то обращение идет к уже загруженной библиотеке. В противном случае поиск идет в следующей последовательности:

1. Директория из которой был запущен модуль, импортирующий dll. Под модулем понимается эксперт, скрипт, индикатор или библиотека EX5;
2. Директория каталог_данных_терминала\MQL5\Libraries (TERMINAL_DATA_PATH\MQL5\Libraries);
3. Директория, из которой был запущен клиентский терминал MetaTrader 5;
4. Системная директория;
5. Директория Windows;
6. Текущая директория;
7. Директории, перечисленные в системной переменной PATH.

Если библиотека DLL использует в своей работе другую DLL, то в случае отсутствия второй DLL первая не сможет загрузиться.

Перед загрузкой эксперта (скрипта, индикатора) формируется общий список всех библиотечных модулей EX5, которые предполагается использовать как из загружаемого эксперта (скрипта, индикатора), так и из библиотек из этого списка. Таким образом обеспечивается однократная загрузка многократно используемых библиотечных модулей EX5. Библиотеки пользуются предопределенными переменными вызвавшего их эксперта (скрипта, индикатора).

Поиск импортируемой библиотеки EX5 производится в следующей последовательности:

1. Директория, путь к которой задается относительно директории импортирующего EX5 эксперта (скрипта, индикатора);
2. Директория каталог_терминала\MQL5\Libraries;
3. Директория MQL5\Libraries в общей директории всех клиентских терминалов MetaTrader 5 (Common\MQL5\Libraries).

Функции, импортируемые из DLL в mql5-программу, должны обеспечивать соглашение о связях, принятое для функций Windows API. Для обеспечения такого соглашения в исходном тексте программ, написанных на языках C или C++, используется ключевое слово __stdcall, которое является специфическим для компиляторов от фирмы Microsoft(r). Обсуждаемое соглашение о связях характеризуется следующим:

© 2000-2025, MetaQuotes Ltd.

- вызывающая функция (в нашем случае mq5-программа) должна "видеть" прототип вызываемой (импортируемой из DLL) функции, для того чтобы правильно сложить параметры на стек;
- вызывающая функция (в нашем случае mql5-программа) складывает параметры на стек в обратном порядке, справа налево - именно в таком порядке импортируемая функция считывает переданные ей параметры;
- параметры передаются по значению, за исключением тех, которые явно передаются по ссылке (в нашем случае строк);
- импортируемая функция, считывая переданные ей параметры, сама очищает стек.

При описании прототипа импортируемой функции можно использовать параметры со значениями по умолчанию.

В случае если соответствующая библиотека не смогла загрузиться, либо установлен запрет на использование DLL, либо импортируемая функция не была найдена – эксперт останавливает свою работу с соответствующим сообщением "expert stopped" в журнале. При этом эксперт не будет запускаться, пока не будет заново проинициализирован. Эксперт может быть переинициализирован в результате перекомпиляции либо после открытия таблицы свойств эксперта и нажатия кнопки OK.

Передача параметров

Все параметры простых типов передаются по значению, если явно не указано, что они передаются по ссылке. При передаче строки передается адрес буфера скопированной строки; если строка передается по ссылке, то в функцию, импортируемую из DLL, передается адрес буфера именно этой строки без копирования.

Структуры, содержащие динамические массивы, строки, классы, другие сложные структуры, а также статические либо динамические массивы перечисленных объектов, не могут быть переданы в качестве параметра в импортируемую функцию.

При передаче в DLL массива всегда (независимо от флага AS_SERIES) передается адрес начала буфера данных. Функция внутри DLL ничего не знает о флаге AS_SERIES, переданный массив является статическим массивом неизвестной длины, для указания размера массива используйте дополнительный параметр.

© 2000-2025, MetaQuotes Ltd.

Ошибки выполнения

В исполняющей подсистеме клиентского терминала существует возможность сохранения кода ошибки в случае ее возникновения при выполнении mql5-программы. Для каждой исполняемой mql5-программы предусмотрена предопределенная переменная _LastError.

Перед запуском функции OnInit переменная _LastError обнуляется. При возникновении ошибочной ситуации во время вычислений или в процессе вызова встроенной функции переменная _LastError принимает соответствующий код ошибки. Значение, сохраненное в этой переменной, можно получить при помощи функции GetLastError().

Существует ряд критических ошибок, при возникновении которых выполнение программы немедленно прерывается:

- деление на ноль;
- выход за пределы массива;
- использование некорректного указателя объекта;

© 2000-2025, MetaQuotes Ltd.

Тестирование торговых стратегий

Идея автоматической торговли привлекательна тем, что торговый робот может без устали работать 24 часа в сутки и семь дней в неделю. Робот не знает усталости, сомнений и страха, ему не ведомы психологические проблемы. Достаточно четко формализовать торговые правила и реализовать их в виде алгоритмов, и робот готов неустанно трудиться. Но прежде необходимо убедиться в соблюдении двух важных условий:

- эксперт совершает торговые операции в соответствии с правилами торговой системы;
- торговая стратегия, реализованная в эксперте, показывает прибыль на истории.

Для получения ответов на эти вопросы предназначен тестер стратегий, входящий в состав клиентского терминала MetaTrader 5.

В данном разделе рассмотрены все особенности тестирования и оптимизации программ в тестере стратегий:

- Ограничения работы функций в тестере торговых стратегий
- Режимы генерации тиков
- Моделирование спреда
- Использование реальных тиков при тестировании
- Глобальные переменные клиентского терминала
- Расчет индикаторов при тестировании
- Загрузка истории при тестировании
- Мультивалютное тестирование
- Моделирование времени в тестере
- Графические объекты при тестировании
- Функция OnTimer() в тестере
- Функция Sleep() в тестере
- Использование тестера для задач оптимизации в математических вычислениях
- Синхронизация баров при тестировании в режиме "Только цены открытия"
- Функция IndicatorRelease() в тестере
- Обработка событий в тестере
- Агенты тестирования
- Обмен данными между терминалом и агентом
- Использование общей папки всех клиентских терминалов
- Использование DLL

Ограничения на память и дисковое пространство в MQL5 Cloud Network

При запуске оптимизации в MQL5 Cloud Network существует ограничение: тестируемый советник не может записывать на диск более 4ГБ информации и использовать более 4ГБ оперативной памяти.

© 2000-2025, MetaQuotes Ltd.

При превышении лимита агент сети не сможет корректно завершить расчет, и вы не получите результат тестирования. При этом с вас будет удержана оплата за уже затраченное на расчеты время.

Если вам необходимо получать информацию с каждого прохода оптимизации, используйте для этого отправку фреймов без записи на диск. То есть при вычислениях в MQL5 Cloud Network не используйте файловые операции в советниках при оптимизации, можно применять такую проверку:

```
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(...);
   ...
  }
```

Ограничения работы функций в тестере торговых стратегий

Существуют ограничения работы некоторых функций в тестере стратегий клиентского терминала.

Функции Comment(), Print() и PrintFormat()

Для увеличения быстродействия при оптимизации параметров советника функции Comment(), Print() и PrintFormat() не выполняются. Исключением является использование этих функций внутри обработчика OnInit(). Это позволяет облегчить поиск причин ошибок при их возникновении.

Функции Alert(), MessageBox(), PlaySound(), SendFTP, SendMail(), SendNotification(), WebRequest()

Функции взаимодействия с "внешним миром" Alert(), MessageBox(), PlaySound(), SendFTP(), SendMail(), SendNotification() и WebRequest() в тестере стратегий не выполняются.

Режимы генерации тиков

Эксперт на языке MQL5 представляет из себя программу, которая запускается каждый раз в ответ на некое внешнее воздействие - событие. Для каждого предопределенного события в эксперте есть соответствующая этому событию функция - обработчик события.

Основным событием для эксперта является изменение цены - NewTick, и поэтому для тестирования экспертов необходимо генерировать тиковые последовательности. В тестере клиентского терминала MetaTrader 5 реализовано 3 режима генерации тиков:

© 2000-2025, MetaQuotes Ltd.

- Все тики
- Цены OHLC с минутных баров (1 Minute OHLC)
- Только цены открытия

Базовым и наиболее детальным режимом генерации является режим "Все тики", остальные два режима являются упрощением основного и будут описаны в сравнении с режимом "Все тики". Рассмотрим все три режима, чтобы понять в чем различие между ними.

Все тики

История котировок по финансовым инструментам передается от торгового сервера в клиентский терминал MetaTrader 5 в виде экономно упакованных блоков минутных баров. Подробную информацию о том, как происходит запроса и построение требуемых таймфреймов можно получить из раздела справки Организация доступа к данным.

Минимальным элементом ценовой истории является минутный бар, из которого можно получить информацию о четырех значениях цены:

- Open - цена, по которой открылся минутный бар;
- High - максимум, который достигался в течение этого минутного бара;
- Low - минимум, который достигался в течение этого минутного бара;
- Close - цена закрытия бара.

Новый минутный бар открывается не в тот момент, когда начинается новая минута (количество секунд становится равным 0), а когда приходит тик - изменение цены хотя бы на один пункт. На рисунке показан первый минутный бар новой торговой недели, который имеет время открытия 2011.01.10. 00:00. Ценовой разрыв между пятницей и понедельником, который мы видим на графике, является обычным явлением, так как даже в выходные дни валютные курсы изменяются в ответ на поступающие новости.

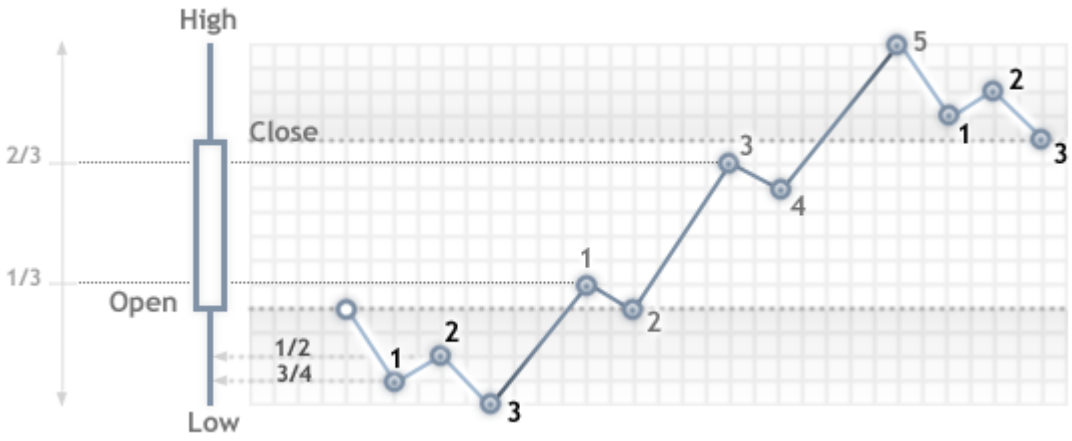


© 2000-2025, MetaQuotes Ltd.

Для этого бара нам только известно, что данный минутный бар открыт 10 января 2011 года в 00 часов 00 минут, но ничего неизвестно о секундах. Это могло быть время 00:00:12 или 00:00:36 (12 или 36 секунд после начала нового дня) или любое другое время в пределах этой минуты. Но мы знаем точно, что в момент открытия нового минутного бара цена Open по EURUSD была на уровне 1.28940.

Точно также мы не знаем с точностью до секунды, когда пришел тик, соответствующий цене закрытия рассматриваемого минутного бара, известно только одно - это последняя цена на минутном баре, которая и была записана как цена Close. Для данной минуты это оказалась цена 1.28958. Время появления цен High и Low также неизвестно, но мы знаем, что максимальная и минимальная цена точно побывала на уровнях 1.28958 и 1.28940 соответственно.

Для тестирования торговой стратегии нам необходима тиковая последовательность, на которой будет эмулироваться работа эксперта. Таким образом, для каждого минутного бара нам известны **4 контрольные точки**, о которых мы точно можем сказать, что цена там побывала. Если бар имеет только 4 тика, то для тестирования этой информации достаточно, но обычно тиковый объем больше 4. Значит, необходимо сгенерировать дополнительные контрольные точки для тиков, которые приходили между ценами Open, High, Low и Close. Принцип генерации тиков в режиме "Все тики" описан в статье Алгоритм генерации тиков в тестере стратегий терминала MetaTrader 5, рисунок из которой представлен ниже.



При тестировании в режиме "Все тики" функция OnTick() эксперта будет **вызываться на каждой контрольной точке**, каждая контрольная точка - это тик из сгенерированной последовательности. Эксперт будет получать время и цену смоделированного тика так же, как и при работе в онлайне.

> **Важно:** режим тестирования "Все тики" является самым точным, но при этом и самым затратным по времени. Для первичной оценки большинства торговых стратегий обычно достаточно использовать один из двух других режимов тестирования.

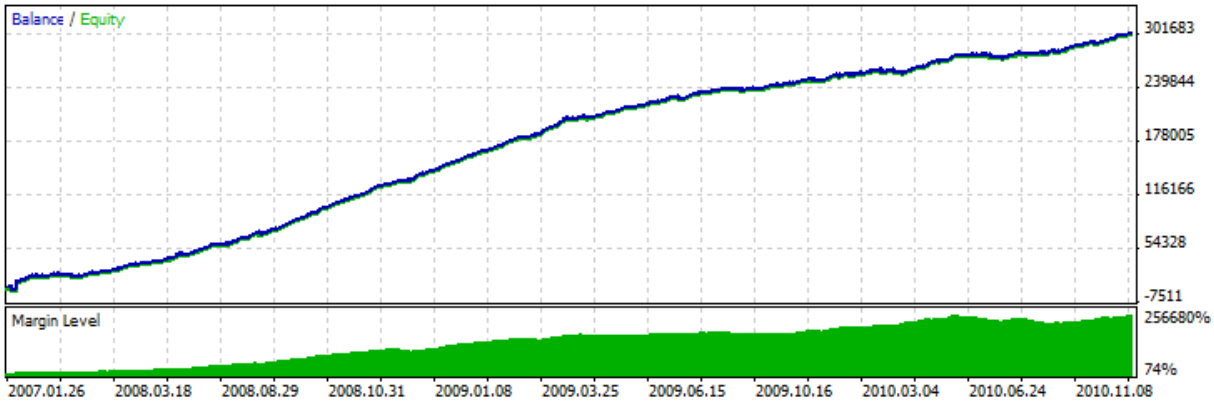
1 minute OHLC

Тестирование в режиме "Все тики" является самым точным из трех режимов, но в то же время и самым медленным. Запуск обработчика OnTick() происходит на каждом тике, а тиковый объем может быть достаточно большим. Для стратегий, которым не важно, в какой тиковой последовательности развивалась цена в течение бара, существует более быстрый и более грубый режим моделирования - "1 minute OHLC".

© 2000-2025, MetaQuotes Ltd.

В режиме "1 minute OHLC" тиковая последовательность строится только по **OHLC ценам минутных баров**, количество сгенерированных контрольных точек существенно уменьшается - следовательно, уменьшается время тестирования. Запуск функции OnTick() производится на всех контрольных точках, которые строятся по ценам OHLC минутных баров.

Отказ от генерации дополнительных промежуточных тиков между ценами Open, High, Low и Close приводит к появлению жесткой детерминированности в развитии цены с того момента, как определена цена Open. Это дает возможность для создания "Грааля тестирования", который показывает красивый восходящий график баланса при тестировании. Пример такого Грааля представлен в CodeBase - Grr-al.



На рисунке представлен очень привлекательный график тестирования этого эксперта. Как он получен? Для минутного бара известно 4 цены, и для них точно известно, что первой идет цена Open, а последней идет цена Close. Между ними есть цены High и Low, последовательность их наступления неизвестна, но известно, что цена High больше или равна цене Open (цена Low меньше или равна цене Open).

Достаточно определить момент поступления цены Open и затем анализировать следующий тик, чтобы определить что перед нами - High или Low. Если цена ниже цены Open, значит, перед нами цена Low - покупаем на этом тике, следующий тик будет соответствовать цене High, на котором закрываем покупку и открываем продажу. Следующий тик последний, это цена Close, на нем закрываем продажу.

Если после цены пришел тик с ценой больше цены открытия, то последовательность сделок обратная. Отработаем в таком мошенническом режиме минутный бар и ждем следующий. При тестировании такого эксперта на истории все идет хорошо, но стоит запустить его в онлайне, и сказка рассыпается - линия баланса по-прежнему ровная, но идет вниз. Для быстрого разоблачения трюка достаточно прогнать такой советник в режиме "Все тики".

Важно: если результаты тестирования эксперта на грубых режимах тестирования ("1 minute OHLC" и "Только цены открытия") слишком хороши, обязательно протестируйте его в режиме "Все тики".

Только цены открытия

В данном режиме происходит генерация тиков по ценам OHLC таймфрейма, выбранного для тестирования. При этом функция эксперта OnTick() запускается только в начале бара по цене Open. Из-за этой особенности стоп-уровни и отложенные ордера могут срабатывать не по

© 2000-2025, MetaQuotes Ltd.

заявленной цене (особенно при тестировании на старших таймфреймах). В обмен за это мы получаем возможность быстро провести оценочное тестирование эксперта.

Исключением при генерации тиков в режиме "Только цены открытия" являются периоды W1 и MN1: для этих таймфреймов тики генерируются для цен OHLC каждого дня, а не для цен OHLC недели или месяца соответственно.

Например, производится тестирование советника на EURUSD H1 в режиме "Только цены открытия". В этом случае общее количество тиков (контрольных точек) будет не больше 4*количество часовых баров, попавших в тестируемый интервал. Но при этом **вызов обработчика OnTick() производится только на открытии** часового **бара**. На остальных ("скрытых" от эксперта) тиках происходят проверки, необходимые для корректного тестирования:

- вычисление маржевых требований;
- срабатывание Stop Loss и Take Profit;
- срабатывание отложенных ордеров;
- удаление отложенных ордеров с истекшим временем.

Если нет открытых позиций или отложенных ордеров, то необходимости в данных проверках на скрытых тиках нет и прирост скорости может оказаться существенным. Данный режим "Только цены открытия" хорошо подходит для тестирования стратегий, которые совершают сделки только на открытии бара и не используют отложенные ордера, а также не используют ордера StopLoss,. TakeProfit. Для класса таких стратегий сохраняется вся необходимая точность тестирования.

В качестве примера эксперта, которому не важно в каком режиме тестироваться, приведем советник Moving Average из стандартной поставки. Логика этого эксперта построена таким образом, что все решения принимаются на открытии бара и сделки проводятся сразу же без использования отложенных ордеров. Запустим тестирование эксперта на EURUSD H1 на интервале с 2010.09.01 по 2010.12.31 и сравним графики. На рисунке показаны графики баланса из отчета тестера для всех трех режимов.



© 2000-2025, MetaQuotes Ltd.

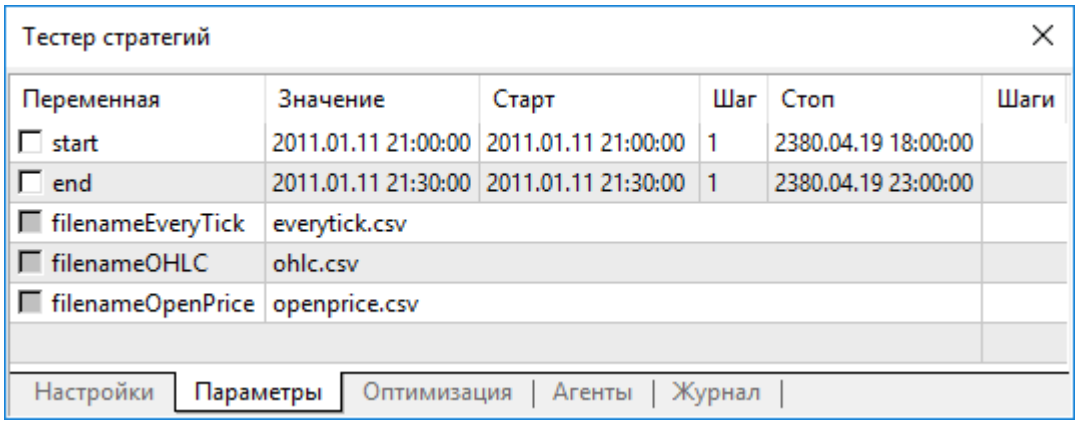
Как видите, графики на разных режимах тестирования абсолютно одинаковы для советника Moving Average из стандартной поставки.

Существует ряд ограничений применения режима "Только цены открытия":

- Нельзя использовать режим торговли "Произвольная задержка";
- В тестируемом эксперте невозможно обратиться к данным более низкого таймфрейма, чем тот, что используется для тестирования/оптимизации. Например, если тестирование/оптимизация осуществляется на периоде H1, то вы можете обращаться к данным H2, H3, H4 и т.д., но не к данным M30, M20, M10 и т.д. Помимо этого, более старшие таймфреймы, к которым идет обращение, должны быть кратными таймфрейму тестирования. Например, при тестировании на периоде M20 нельзя обратиться к таймфрейму M30, но можно к H1. Эти ограничения обусловлены невозможностью получить данные более низких и не кратных таймфреймов из баров, генерируемых при тестировании/оптимизации.
- Ограничения по обращению к данным других таймфремов распространяются и на другие символы, чьи данные используются советником. Однако в этом случае ограничением для каждого символа служит первый таймфрейм, к которому произошло обращение во время тестирования/оптимизации. Например, тестирование осуществляется на символе и периоде EURUSD H1, советник в первый раз обратился к символу GBPUSD M20. В этой ситуации советник в дальнейшем может использовать данные EURUSD H1, H2, и т.д., а также GBPUSD M20, H1, H2 и т.д.

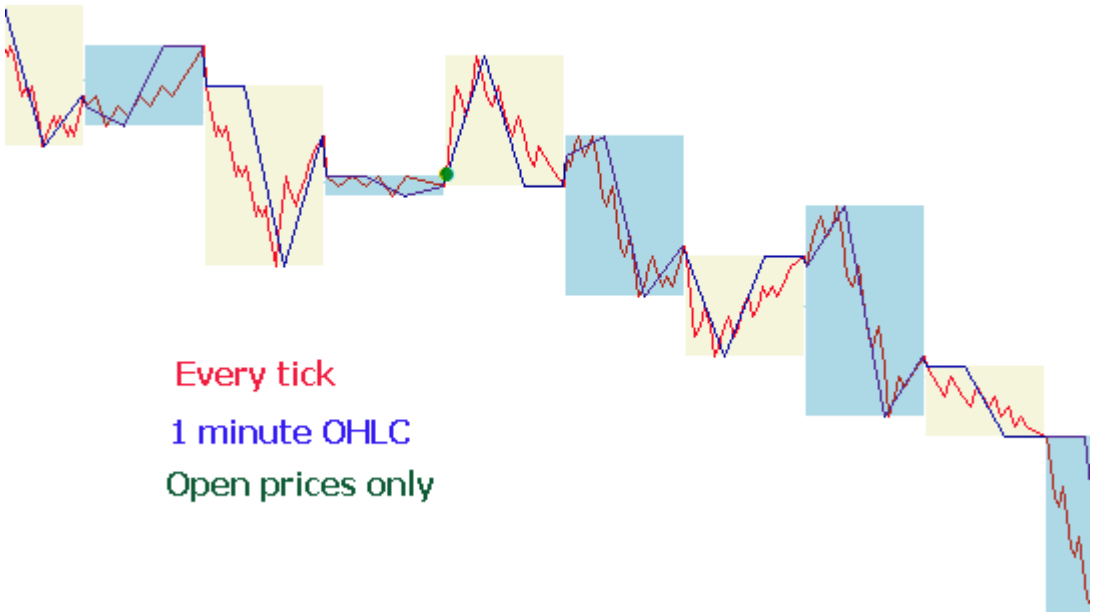
Важно: режим "Только цены открытия" является самым быстрым по времени тестирования, но подходит не для всех торговых стратегий. Выбирайте необходимый режим тестирования исходя из особенностей работы торговой системы.

В завершение раздела о режимах моделирования приведем визуальное сравнение разных режимов генерации тиков для EURUSD для двух баров M15 на интервале с 2011.01.11 21:00:00 - 2011.01.11 21:30:00. Запись тиков произведена с помощью эксперта WriteTicksFromTester.mq5 в разные файлы, окончание имен этих файлов задаются input-параметрах filenameEveryTick, filenameOHLC и filenameOpenPrice.



Чтобы получить три файла с тремя тиковыми последовательностями (для каждого из режимов "Все тики", "OHLC на минутных барах" и "Только цены открытия") советник был запущен трижды в соответствующих режимах на одиночных прогонах. Затем данные из этих трех файлов с помощью индикатора TicksFromTester.mq5 были выведены на график. Код индикатора прилагается к статье.

© 2000-2025, MetaQuotes Ltd.



По умолчанию все файловые операции в языке MQL5 производятся в пределах "файловой песочницы" и при тестировании эксперту доступна только собственная "файловая песочница". Для того чтобы индикатор и эксперт при тестировании работали с файлами из одной папки, использовался флаг FILE_COMMON. Пример кода из эксперта:

```
//--- откроем файл
   file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- проверим успешность операции
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Не удалось открыть на запись файл %s. Код ошибки=%d",filename,GetLa
      return;
     }
   else
     {
      //--- сообщим о записи в общую папку всех клиентских терминалов и ее местоположе
      PrintFormat("Файл будет записан в папке %s",TerminalInfoString(TERMINAL_COMMONDA
     }
```

В индикаторе для чтения данных также использовался флаг FILE_COMMON, это позволило избежать переноса необходимых файлов вручную из одной папки в другую.

```
//--- откроем файл
   int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//--- проверим успешность операции
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Не удалось открыть на чтение файл %s. Код ошибки=%d",fname,GetLastE
      return;
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
   {
    //--- сообщим местонахождение общей папки всех клиентских терминалов
    PrintFormat("Файл будет прочитан из папки %s",TerminalInfoString(TERMINAL_COMMON
   }
```

Моделирование спреда

Разница между ценами Bid и Ask называется спредом. При тестировании спред не моделируется, а берется из исторических данных. Если в исторических данных спред меньше или равен нулю, то используется последний известный на момент генерации спред.

В тестере спред всегда считается плавающим. То есть SymbolInfoInteger(symbol, SYMBOL_SPREAD_FLOAT) всегда возвращает true.

Кроме того, в исторических данных хранятся значения тиковых и торговых объемов. Для хранения и получения данных используется специальная структура MqlRates:

```
struct MqlRates
  {
   datetime time;         // время открытия бара
   double   open;         // цена открытия Open
   double   high;         // наивысшая цена High
   double   low;          // наименьшая цена Low
   double   close;        // цена закрытия Close
   long     tick_volume;  // тиковый объем
   int      spread;       // спред
   long     real_volume;  // биржевой объем
  };
```

Использование реальных тиков при тестировании

Тестирование и оптимизация на реальных тиках являются максимально приближенными к реальным условиям. Вместо сгенерированных на основе минутных данных используются реальные тики, накопленные по финансовым инструментам брокером. Это — тики с бирж и от поставщиков ликвидности.

Чтобы обеспечить наибольшую точность при тестировании, в режиме реальных тиков также используются и минутные бары. По ним проверяются и корректируются тиковые данные. Это также позволяет избежать расхождения графиков в тестере и клиентском терминале.

Тестер проверяет соответствие тиковых данных параметрам минутного бара: тик не должен выходить за пределы цен High/Low бара, открывающий и закрывающий минуту тик должен совпадать с ценами Open/Close бара. Также сравнивается объем. При выявлении несовпадения отбрасываются все тики, соответствующие этому минутному бару. Вместо них будут использованы сгенерированные тики (как в режиме "Все тики").

Если в истории символа есть минутный бар, но тиковых данных за эту минуту нет, тестер сгенерирует тики в режиме "Все тики". Это позволяет выстроить правильный график в тестере в случае неполных тиковых данных у брокера.

© 2000-2025, MetaQuotes Ltd.

Если в истории символа нет минутного бара, но тиковые данные за эту минуту есть, они могут быть использованы в тестере. Например, бары биржевых символов формируются по ценам Last. Если с сервера приходят только тики с ценами Bid/Ask без цены Last, бар не будет сформирован. Тестер будет использовать эти тиковые данные, поскольку они не противоречат минутным.

Тиковые данные могут не совпадать с минутными барами по различным причинам. Например, из-за обрывов связи или иных сбоев при передаче данных от источника в клиентский терминал. При тестировании минутные данные считаются более достоверными.

При тестировании на реальных тиках учитывайте следующие особенности:

- При запуске тестирования синхронизируются не только тиковые, но и минутные данные по инструменту.
- Тики хранятся в кэше символа в тестере стратегий. Размер кэша — не более 128 000 тиков. При поступлении новых тиков самые старые данные из него выталкиваются. Однако при помощи функции CopyTicks можно получить тики и за пределами кэша (только при тестировании по реальным тикам). В этом случае данные будут запрошены из базы тиков тестера, которая полностью соответствует аналогичной базе клиентского терминала. В эту базу никакие корректировки по минутным барам не вносятся. Поэтому тики в ней могут отличаться от тиков, находящихся в кэше.

Глобальные переменные клиентского терминала

При тестировании глобальные переменные клиентского терминала также эмулируются, но они никак не связаны с настоящими глобальным переменным терминала, которые можно увидеть в терминале по кнопке F3. Это означает, что все операции с глобальными переменными терминала при тестировании производятся вне самого клиентского терминала (в агенте тестирования).

Расчет индикаторов при тестировании

В режиме реального времени значения индикаторов вычисляются на каждом тике.

В тестере стратегий индикаторы рассчитываются только при обращении к ним за данными — то есть только в тот момент, когда запрашиваются значения индикаторных буферов. Исключение составляют пользовательские индикаторы с выставленным #property tester_everytick_calculate, в этом случае пересчет идет на каждом тике.

В визуальном режиме тестирования все индикаторы пересчитываются безусловно при приходе нового тика, для того чтобы правильно отображаться на визуальном графике тестирования.

Расчет индикатора на каждом тике делается однократно, и все последующие обращения за данными индикатора до поступления нового тика не вызывают пересчета. Поэтому, если в эксперте с помощью функции EventSetTimer() включен таймер, то перед каждым вызовом обработчика OnTimer() будут запрошены данные индикатора с момента последнего тика. Если на последнем тике расчет индикатора еще не производился, то будут запущены вычисления значений индикатора. Если данные уже были подготовлены, то они будут предоставлены без нового пересчета.

Таким образом, все расчеты индикаторов делаются максимально экономно — если на данном тике индикатор уже был рассчитан, то данные индикатора отдаются как есть, повторный расчет индикатора не запускается.

© 2000-2025, MetaQuotes Ltd.

Загрузка истории при тестировании

История по тестируемому инструменту синхронизируется и закачивается терминалом с торгового сервера перед запуском процесса тестирования. При этом в первый раз терминал скачивает с торгового сервера сразу всю доступную по тестируемому инструменту историю, чтобы впоследствии не обращаться за ней. В дальнейшем происходит лишь докачка новых данных.

Агент тестирования получает от клиентского терминала историю по тестируемому инструменту сразу же после запуска тестирования. Если в процессе тестирования используются данные по другим инструментам (например, это мультивалютный эксперт), то в этом случае агент тестирования запрашивает у клиентского терминала требуемую историю при первом же обращении. Если исторические данные имеются на терминале, они сразу передаются на агенты тестирования. Если данные отсутствуют, терминал запросит и скачает их с сервера, а затем передаст на агенты тестирования.

Обращение к дополнительным инструментам происходит и в том случае, когда вычисляется цена кросс-курса при торговых операциях. Например, при тестировании стратегии на EURCHF с валютой депозита в долларах США перед обработкой первой же торговой операцией агент тестирования запрашивает у клиентского терминала историю по EURUSD и USDCHF, хотя в стратегии нет прямого обращения к этим инструментам.

Перед началом тестирования мультивалютной стратегии рекомендуется предварительно скачать все необходимые исторические данные на клиентском терминале. Это позволит избежать задержек при тестировании/оптимизации, связанных с докачкой данных. Закачать историю можно, например, путем открытия соответствующих графиков и прокрутки их к началу истории. Пример принудительной загрузки истории в торговый терминал приведен в документации по MQL5 в разделе Организация доступа к данным.

Тестерные агенты в свою очередь получают историю от терминала и также в упакованном виде. При повторном тестировании загрузка тестером истории из терминала уже не происходит, потому что данные есть от предыдущего запуска тестера.

- Терминал загружает историю с торгового сервера только один раз при первом обращении агента к терминалу за историей для тестируемого символа. История загружается в упакованном виде, чтобы сократить трафик.
- Тики не пересылаются по сети, они генерируются на тестерных агентах.

Мультивалютное тестирование

Тестер позволяет проводить проверку на истории стратегий, торгующих на нескольких инструментах. Такие эксперты условно называют мультивалютными, так как изначально в предыдущих платформах тестирование проводилось только для одного инструмента. В тестере же терминала MetaTrader 5 можно моделировать торговлю по всем доступным инструментам.

История по используемым инструментам закачивается тестером из **клиентского терминала** (не с торгового сервера!) автоматически при первом обращении к данному инструменту.

Агент тестирования закачивает только недостающую историю с небольшим запасом, чтобы обеспечить необходимые данные на истории для расчета индикаторов на момент начала тестирования. Минимальный объем истории при скачивании с торгового сервера для таймфреймов

© 2000-2025, MetaQuotes Ltd.

D1 и меньше составляет один год. Таким образом, если запускается тестирование на интервале 2010.11.01-2010.12.01 (тестирование на интервале в один месяц) с периодом M15 (каждый бар равен 15 минутам), то у терминала будет запрошена история по инструменту за весь 2010 год. Для таймфреймов Weekly будет запрошена история в 100 баров, что составляет примерно два года (в году 52 недели). Для тестирования на месячном таймфрейме Monthly агент запросит историю за 8 лет (12 месяцев * 8 лет = 96 месяцев).

Если не удается по каким-либо причинам обеспечить необходимое количество баров перед началом тестирования, то **дата начала будет автоматически сдвинута** от прошлого к настоящему для того чтобы обеспечить такое количество баров.

При тестировании эмулируется также и "Обзор рынка", из которого можно получать информацию по инструментам. По умолчанию в начале тестирования в "Обзоре рынка" тестера есть только один символ - символ на котором запущено тестирование. Все необходимые символы подключаются к "Обзору рынка" тестера (не терминала!) автоматически при обращении к ним.

> Перед началом тестирования мультивалютного эксперта необходимо выбрать требуемые для тестирования инструменты в "Обзоре рынка" терминала и подкачать данные на нужную глубину. При первом же обращении к "чужому" символу будет автоматически произведена синхронизация по этому символу между агентом тестирования и клиентским терминалом. "Чужой" символ - это символ, отличающийся от того, на котором запущено тестирование.

Обращение к данным чужого символа происходят в следующих случаях:

- использование функций технических индикаторов и IndicatorCreate() на паре символ/таймфрейм;
- запрос к "Обзору рынке" (Market Watch) по чужому символу:
  1. SeriesInfoInteger
  2. Bars
  3. SymbolSelect
  4. SymbolIsSynchronized
  5. SymbolInfoDouble
  6. SymbolInfoInteger
  7. SymbolInfoString
  8. SymbolInfoTick
  9. SymbolInfoSessionQuote
  10. SymbolInfoSessionTrade
  11. MarketBookAdd
  12. MarketBookGet
- запрос к таймсерии по паре символ/период функциями:
  1. CopyBuffer
  2. CopyRates
  3. CopyTime
  4. CopyOpen
  5. CopyHigh

© 2000-2025, MetaQuotes Ltd.

6. CopyLow
7. CopyClose
8. CopyTickVolume
9. CopyRealVolume
10. CopySpread

В тот момент, когда происходит первое обращение к чужому символу, процесс тестирования останавливается и происходит подкачка истории по паре символ/период от терминала к агенту тестирования. Одновременно включается генерация тиковой последовательности для этого символа.

Для каждого инструмента генерируется собственная тиковая последовательность в соответствие с выбранным режимом генерации тиков. Кроме того, можно явно запросить историю для нужных символов с помощью вызова функции SymbolSelect() в обработчике OnInit() - загрузка истории будет произведена сразу же до начала тестирования советника.

Таким образом, для проведения мультивалютного тестирования в клиентском терминале MetaTrader 5 не требуется предпринимать никаких дополнительных усилий. Достаточно открыть графики соответствующих инструментов в клиентском терминале. История по нужным символам будет автоматически загружена с торгового сервера при условии, что эти данные есть на нем.

Моделирование времени в тестере

При тестировании локальное время TimeLocal() всегда равно серверному времени TimeTradeServer(). В свою очередь, серверное время всегда равно времени, соответствующему времени GMT - TimeGMT(). Таким образом, все эти функции при тестировании выдают одно и то же время.

Отсутствие разницы между GMT, локальным и серверным временем в тестере сделано сознательно по той самой причине, что связь с сервером может быть не всегда. А результаты тестирования должны быть одинаковыми, независимо от наличия связи. Информация о серверном времени не хранится локально, а берётся с сервера.

Графические объекты при тестировании

Во время тестирования/оптимизации не осуществляется построение графических объектов. Таким образом, при обращении к свойствам созданного объекта во время тестирования/оптимизации эксперт получит нулевые значения.

> Данное ограничение не распространяется на тестирование в визуальном режиме.

Функция OnTimer() в тестере

В MQL5 возможна обработка событий таймера. Вызов обработчика OnTimer() производится независимо от режима тестирования. Это означает, что если тестирование запущено в режиме "Только цены открытия" на периоде H4 и внутри эксперта установлен таймер с вызовом каждую секунду, то на открытии каждого H4 бара один раз будет вызван обработчик OnTick() и 14400 раз (3600 секунд * 4 часа) в течение бара будет вызван обработчик OnTimer(). Насколько при этом увеличится время тестирования эксперта, зависит от логики эксперта.

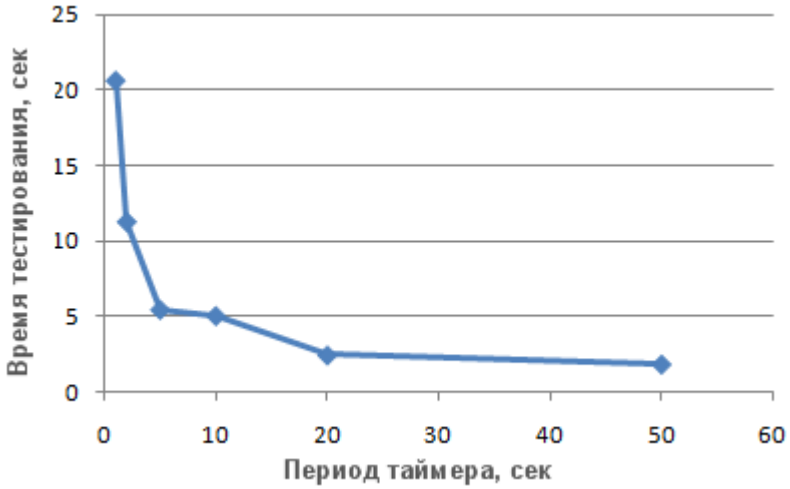
© 2000-2025, MetaQuotes Ltd.

Для проверки зависимости времени тестирования от заданной периодичности таймера был написан простой эксперт без торговых операций.

```
//--- input parameters
input int      timer=1;              // значение таймера, сек
input bool     timer_switch_on=true; // таймер включен
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- запустим таймер если  timer_switch_on==true
   if(timer_switch_on)
     {
      EventSetTimer(timer);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- остановим таймер
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
// ничего не делаем, тело обработчика пустое
  }
//+------------------------------------------------------------------+
```

Были сделаны замеры времени тестирования при различных значениях параметра timer (периодичность события Timer). На полученных данных построен график зависимости времени тестирования T от значения периодичности Period.

© 2000-2025, MetaQuotes Ltd.



Хорошо видно, чем меньше параметр timer при инициализации таймера функцией EventSetTimer(timer), тем меньше период (Period) между вызовами обработчика OnTimer(), и тем больше время тестирования T при одних и тех же остальных условиях.

Функция Sleep() в тестере

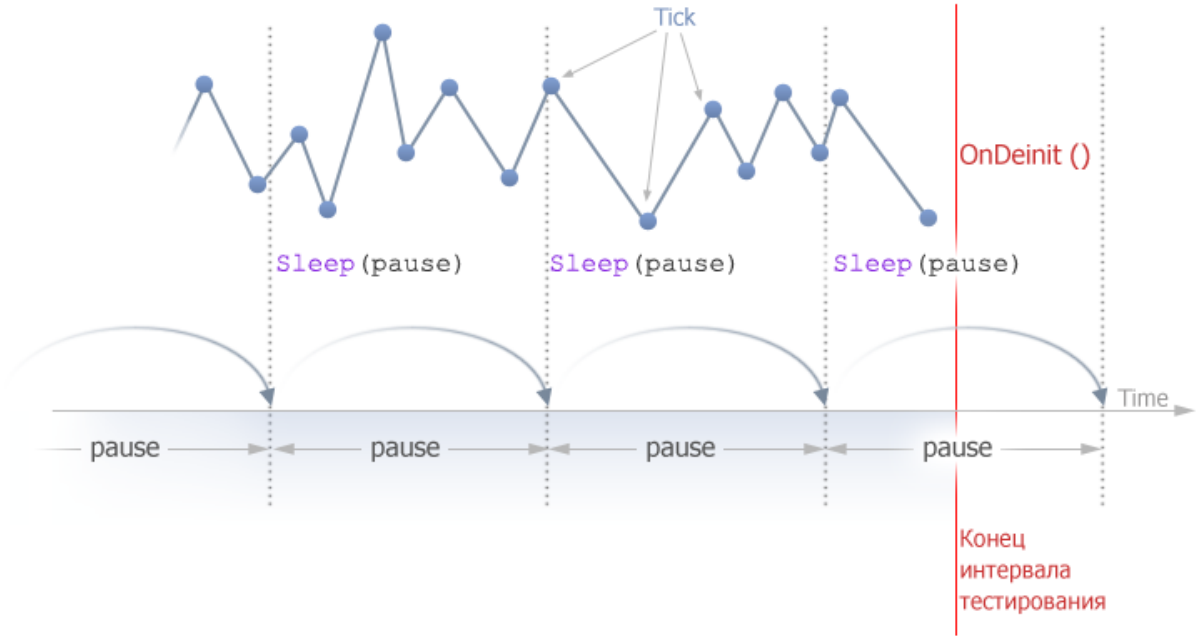
Функция Sleep() позволяет в эксперте или скрипте приостановить выполнение mql5-программы на некоторое время при работе на графике. Это может понадобиться при запросе каких-либо данных, которые в момент запроса еще не готовы и необходимо дождаться момента их готовности. Подробный пример использования функции Sleep() можно посмотреть в разделе Организация доступа к данным.

В тестере же вызовы Sleep() не задерживают процесс тестирования. При вызове Sleep() "проигрываются" сгенерированные тики в пределах указанной задержки, в результате чего могут сработать отложенные ордера, стопы и т.д. После вызова Sleep() смоделированное в тестере время увеличивается на интервал, указанный в параметре функции Sleep.

Если в результате выполнения функции Sleep() текущее время в тестере вышло за конец периода тестирования, то будет получена ошибка "бесконечный цикл в Sleep". При получение такой ошибки результаты тестирования не отбрасываются, все вычисления производятся в полном объеме (количество сделок, просадка и т.д.) и результаты данного тестирования передаются терминалу.

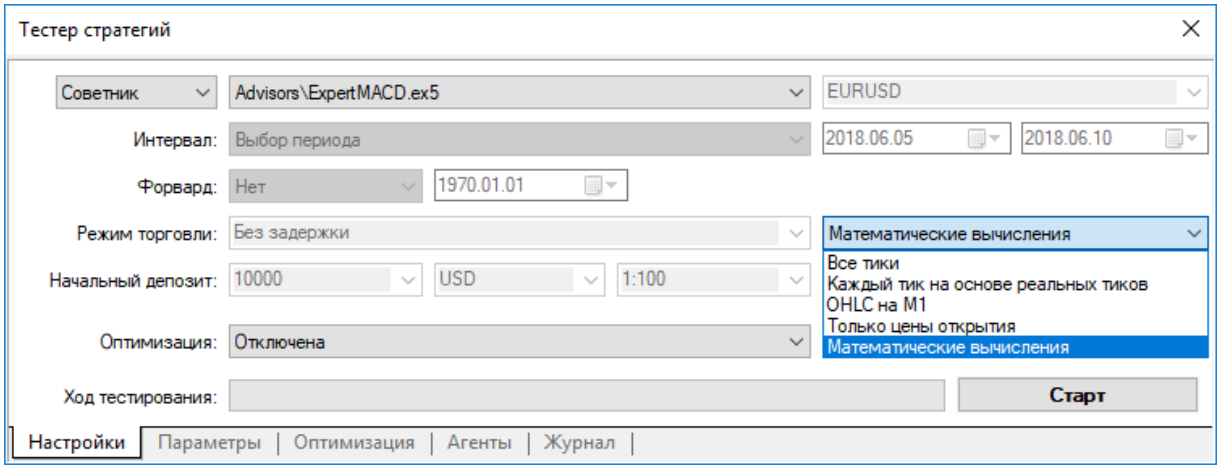
Функция Sleep() не будет работать в OnDeinit(), так как после ее вызова тестерное время гарантированно окажется за пределами интервала тестирования.

© 2000-2025, MetaQuotes Ltd.



Использование тестера для задач оптимизации в математических вычислениях

Тестер в терминале MetaTrader 5 можно использовать не только для проверки торговых стратегий, но и для математических расчётов. Для этого необходимо выбрать соответствующий режим в настройках:



При выборе режима "Математические вычисления" будет произведен "пустой" прогон агента тестирования. Пустой прогон означает, что не будет производиться генерация тиков и загрузки истории. При таком прогоне будут вызваны только три функции: OnInit(), OnTester(), OnDeinit().

Если дата окончания тестирования меньше или равна дате начала тестирования, то это также будет означать тестирования в режиме "Математические вычисления".

При использовании тестера для решения математических задач закачка истории и генерация тиков не происходят.

© 2000-2025, MetaQuotes Ltd.

Типичная математическая задача для решения в тестере MetaTrader 5 - поиск экстремума от функции многих переменных. Для ее решения необходимо:

- Разместить блок вычислений значения функции от многих переменных в OnTester() и вернуть вычисленное значение через return(значение_функции);
- Вынести параметры функции в глобальную область программы в виде input-переменных;

Компилируем советник, открываем окно "Тестер". На вкладке "Входные параметры" отмечаем требуемые входные переменные и задаем для них задаем границы в пространстве значений и шаг для перебора.

Выбираем тип оптимизации - "Медленный (полный перебор параметров) или "Быстрая (генетический алгоритм)". Для простого поиска экстремума функции лучше выбрать быструю оптимизацию, но если нужно вычислить значения на всем пространстве переменных, то подойдет медленная оптимизация.

Выбираем режим "Математические вычисления" и запускаем кнопкой "Старт" процедуру оптимизации. Необходимо помнить, что при оптимизации всегда ищется локальный максимум значения функции OnTester. Для поиска локального минимума можно из функции OnTester возвращать значение, обратное вычисленному значению функции:

```
return(1/значение_функции);
```

При этом необходимо самостоятельно реализовать проверку, чтобы значение_функции не было равно нулю, так как в противном случае можно получить критическую ошибку деления на ноль. Есть и другой вариант, более подходящий и не искажающий результаты оптимизации, он предложен читателями статьи:

```
return(-значение_функции);
```

В этом варианте не требуется проверять значение_функции на равенство нулю и сама поверхность результатов оптимизации в 3D-представлении имеет ту же форму, только зеркально отраженную от исходной.

В качестве примера приведем функцию sink():

$$sink(x,y) = \sin(x^2 + y^2)$$

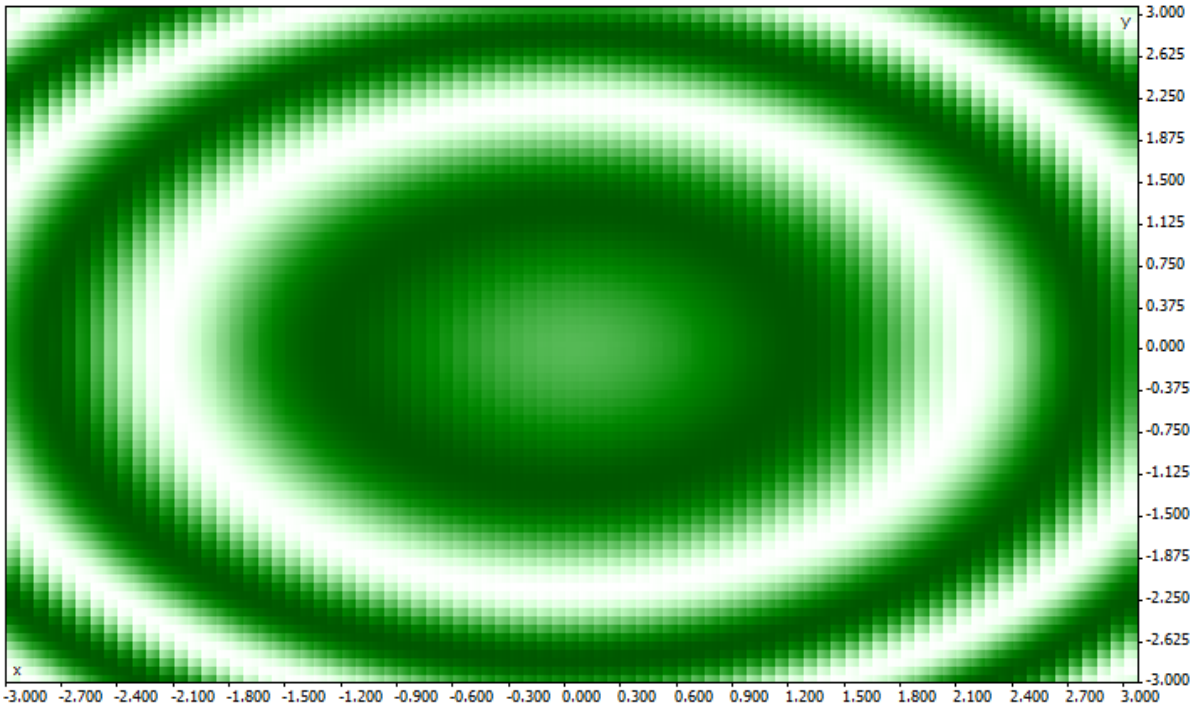
Код советника для поиска экстремума этой функции поместим в OnTester():

```
//+------------------------------------------------------------------+
//|                                                         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"
//--- input parameters
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                                                  |
```

© 2000-2025, MetaQuotes Ltd.

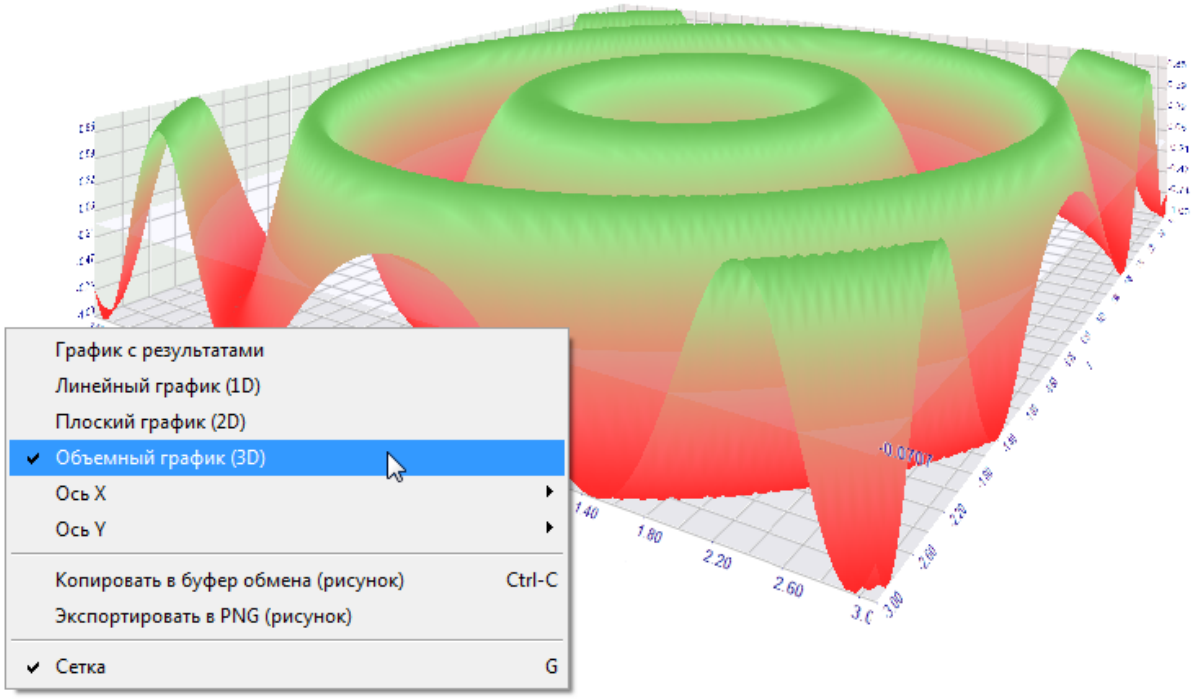
```
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   double sink=MathSin(x*x+y*y);
//---
   return(sink);
  }
//+------------------------------------------------------------------+
```

Проведем оптимизацию и представим результаты оптимизации в виде 2D графика.



Чем лучше значение для заданной пары параметров (x,y), тем более насыщенный цвет. Как и ожидалось из вида формулы функции sink(), ее значения образуют концентрические круги с центром в точке (0,0). Для функции sink() не существует абсолютного экстремума. Это хорошо видно при просмотре результатов оптимизации в режиме 3D:

© 2000-2025, MetaQuotes Ltd.



Синхронизация баров при тестировании в режиме "Только цены открытия"

Тестер в клиентском терминале MetaTrader 5 позволяет проверять и, так называемые, "мультивалютные" советники. Мультивалютный советник - это советник, который торгует на двух или более символах.

Тестирование стратегий, торгующих на нескольких инструментах, налагает на тестер несколько дополнительных технических требований:

- генерации тиков для этих инструментов;
- расчет значений индикаторов для этих инструментов;
- расчет маржевых требований по этим инструментам;
- синхронизация сгенерированных тиковых последовательностей по всем торгуемым инструментам.

Тестер генерирует и проигрывает для каждого инструмента тиковую последовательность в соответствии с выбранным режимом торговли. При этом новый бар на каждом инструменте открывается независимо от того, как открылся бар на другом инструменте. Это означает, что при тестировании мультивалютного эксперта возможна ситуация (и чаще всего так и бывает), когда на одном инструменте новый бар уже открылся, а на другом еще нет. Таким образом, при тестировании все происходит как в жизни.

Такое достоверное моделирование развития истории в тестере не вызывает вопросов до тех пор, пока используются режимы тестирования "Все тики" и "1 minute OHLC". При этих режимах в пределах одной свечи генерируется достаточное количество тиков, чтобы дождаться момента синхронизации баров с разных символов. Но как тестировать мультивалютные стратегии в режиме "Только цены открытия", если требуется обязательная синхронизация баров на торгуемых

© 2000-2025, MetaQuotes Ltd.

инструментах? Ведь в этом режиме эксперт вызывается только на одном тике, который соответствует времени открытия бара.

Поясним на примере: если мы тестируем эксперта на символе EURUSD, и на EURUSD открылась новая часовая свеча, то мы легко узнаем этот факт - при тестировании в режиме "Только цены открытия" событие NewTick соответствует моменту открытия бара на тестируемом периоде. Но при этом нет никакой гарантии, что новая свеча открылась по символу GBPUSD, который используется в эксперте.

В обычных условиях достаточно завершить работу функции OnTick() и проверить появление нового бара на GBPUSD на следующем тике. Но при тестировании в режиме "Только цены открытия" другого тика не будет, и может показаться, что этот режим не годится для тестирования мультивалютных экспертов. Но это не так - не забывайте, что тестер в MetaTrader 5 ведет себя также, как и в жизни. Можно дождаться момента, когда на другом символе откроется новый бар с помощью функции Sleep()!

Код советника Synchronize_Bars_Use_Sleep.mq5, который демонстрирует пример синхронизации баров при тестировании в режиме "Только цены открытия":

```
//+------------------------------------------------------------------+
//|                                  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"
//--- input parameters
input string   other_symbol="USDJPY";
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- сверим текущий символ
   if(_Symbol==other_symbol)
     {
      PrintFormat("Необходимо указать другой символ или запустить тестирование на дру
      //--- принудительно прекращаем тестирование эксперта
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- статическая переменная для хранения времени открытия последнего бара
```

© 2000-2025, MetaQuotes Ltd.

```
   static datetime last_bar_time=0;
//---  признак того, что время открытия последнего бара на разных символах синхронизир
   static bool synchonized=false;
//--- если статическая переменная еще неинициализирована
   if(last_bar_time==0)
     {
      //--- это первый вызов, запишем время открытия и выйдем
      last_bar_time=(datetime)SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
      PrintFormat("Инициализировали переменную last_bar_time значением %s",TimeToStrin
     }
//--- получим время открытия последнего бара по своему символу
   datetime curr_time=(datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DAT
//--- если время открытия текущего бара не совпадает с тем, что хранится в last_bar_ti
   if(curr_time!=last_bar_time)
     {
      //--- запомним время открытия нового бара в статической переменной
      last_bar_time=curr_time;
      //--- синхронизация нарушена, сбросим флаг в false
      synchonized=false;
      //--- выведем сообщение об этом событии
      PrintFormat("На символе %s открылся новый бар в %s",_Symbol,TimeToString(TimeCur
     }
//--- сюда будем сохранять время открытия бара на чужом символе
   datetime other_time;
//--- цикл, пока время открытия последнего бара по другому символу не сравняется с cur
   while(!(curr_time==(other_time=(datetime)SeriesInfoInteger(other_symbol,Period(),SE
     {
      PrintFormat("Подождем 5 секунд..");
      //--- подождем 5 секунд и опять запросим SeriesInfoInteger(other_symbol,Period()
      Sleep(5000);
     }
//--- время открытия бара теперь одинаково для обоих символов
   synchonized=true;
   PrintFormat("Время открытия последнего бара на своем символе %s: %s",_Symbol,TimeTo
   PrintFormat("Время открытия последнего бара на символе %s: %s",other_symbol,TimeToS
//--- TimeCurrent() не подойдет, используем TimeTradeServer() для
   Print("Бары синхронизировались в ",TimeToString(TimeTradeServer(),TIME_DATE|TIME_SE
  }
//+------------------------------------------------------------------+
```

Обратите внимание на последнюю строчку в советнике, которая выводит текущее время, когда был установлен факт синхронизации:

```
   Print("Бары синхронизировались в ",TimeToString(TimeTradeServer(),TIME_SECONDS));
```

Для вывода текущего времени мы использовали функцию TimeTradeServer(), а не TimeCurrent(). Дело в том, что функция TimeCurrent() возвращает время последнего тика, которое никак не изменилось после использования Sleep(). Запустите советник в режиме "Только цены открытия" и увидите сообщения о синхронизации баров.

© 2000-2025, MetaQuotes Ltd.

Core 1	2010.12.01 20:00:05	Бары синхронизировались в 2010.12.01 20:00:05
Core 1	2010.12.01 20:00:05	Время открытия последнего бара на символе USDJPY: 2010.12.01 20:00
Core 1	2010.12.01 20:00:05	Время открытия последнего бара на своем символе EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 20:00:00	Подождем 5 секунд..
Core 1	2010.12.01 20:00:00	На символе EURUSD открылся новый бар в 2010.12.01 20:00
Core 1	2010.12.01 16:00:05	Бары синхронизировались в 2010.12.01 16:00:05

Используйте функцию TimeTradeServer() вместо TimeCurrent(), если требуется получить текущее серверное время, а не время поступления последнего тика.

Есть и другой способ синхронизации баров - с помощью таймера. Пример такого эксперта Synchronize_Bars_Use_OnTimer.mq5 приложен к статье.

Функция IndicatorRelease() в тестере

После окончания одиночного тестирования автоматически открывается график инструмента, на котором отображаются совершенные сделки и индикаторы, которые использовались в эксперте. Это помогает визуально проверить моменты входа и выхода, а также сопоставить их со значениями индикаторов.

> **Важно:** индикаторы, отображаемые на автоматически открытом после завершения тестирования графике, рассчитываются заново уже после окончания тестирования. Даже если эти индикаторы использовались в тестируемом эксперте.

Но в некоторых случаях программисту может понадобиться скрыть информацию о том, какие индикаторы задействованы в торговом алгоритме. Например, код эксперта сдается в аренду или продается в виде исполняемого файла без предоставления исходного кода. Для этих целей подойдет функция IndicatorRelease().

Если в терминале задан шаблон с названием tester.tpl в каталоге /profiles/templates клиентского терминала, то именно он будет применен к открываемому графику. При его отсутствии применяется шаблон по умолчанию (default.tpl).

Функция IndicatorRelease() изначально предназначена для освобождения расчетной части индикатора, если он больше не нужен. Это позволяет экономить как память, так и ресурсы процессора, потому что каждый тик вызывает расчет индикатора. Второе ее предназначение - запретить показ индикатора на графике тестирования после окончания одиночного прогона.

Чтобы запретить показ индикатора на графике по окончании тестирования, вызовете IndicatorRelease() с хэндлом индикатора в обработчике OnDeinit(). Функция OnDeinit() всегда вызывается после завершения и перед показом графика тестирования.

```
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   bool hidden=IndicatorRelease(handle_ind);
   if(hidden)
      Print("IndicatorRelease() выполнена успешно");
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      Print("IndicatorRelease() вернула false. Код ошибки ",GetLastError());
  }
```

Для того чтобы запретить показ индикатора на графике после завершения одиночного тестирования, используйте функцию IndicatorRelease() в обработчике OnDeinit().

Обработка событий в тестере

Наличие обработчика OnTick() в эксперте не является обязательным для того чтобы его можно было подвергнуть проверке на исторических данных в тестере терминала MetaTrader 5. Достаточно того, чтобы в советнике была хотя бы одна функция-обработчик из перечисленных:

- OnTick() - обработчик события прихода нового тика;
- OnTrade() - обработчик торгового события;
- OnTimer() - обработчик события прихода сигнала от таймера;
- OnChartEvent() - обработчик пользовательских событий.

При тестировании в эксперте можно обрабатывать пользовательские события с помощью функции OnChartEvent(), но в индикаторах эта функция в тестере не вызывается. Даже если индикатор имеет обработчик OnChartEvent() и этот индикатор используется в тестируемом эксперте, то сам индикатор не будет получать никаких пользовательских событий.

Индикатор при тестировании может генерировать пользовательские события с помощью функции EventChartCustom(), а советник может обрабатывать это событие в OnChartEvent().

Помимо вышеуказанных событий в тестере стратегий генерируются специальные события, связанные с процессом тестирования и оптимизации:

- Tester - данное событие генерируется по окончании тестирования эксперта на исторических данных. Обработка события Tester производится функцией OnTester(). Эта функция может быть использована только в экспертах при тестировании и предназначена в первую очередь для расчета некоторого значения, используемого в качестве критерия Custom max при генетической оптимизации входных параметров.
- TesterInit - данное событие генерируется при запуске оптимизации в тестере стратегий перед самым первым проходом. Обработка события TesterInit производится функцией OnTesterInit(). Эксперт, имеющий данный обработчик, при запуске оптимизации автоматически загружается на отдельном графике терминала с указанными в тестере символом и периодом, и получает событие TesterInit. Функция предназначена для инициализации эксперта перед началом оптимизации для последующей обработки результатов оптимизации.
- TesterPass - данное событие генерируется при поступлении нового фрейма данных. Обработка события TesterPass производится функцией OnTesterPass(). Эксперт с данным обработчиком автоматически загружается на отдельном графике терминала с указанными для тестирования символом/периодом и получает во время оптимизации события TesterPass при получении фрейма. Функция предназначена для динамической обработки результатов оптимизации прямо "на лету", не дожидаясь её окончания. Добавление фреймов производится функцией FrameAdd(), которую можно вызывать по окончании одиночного прохода в обработчике OnTester().
- TesterDeinit - данное событие генерируется по окончании оптимизации эксперта в тестере стратегий. Обработка события TesterDeinit производится функцией OnTesterDeinit(). Эксперт с данным обработчиком автоматически загружается на график при запуске оптимизации и

© 2000-2025, MetaQuotes Ltd.

получает событие TesterDeinit после её завершения. Функция предназначена для финальной обработки всех результатов оптимизации.

Агенты тестирования

Тестирование в клиентском терминале MetaTrader 5 осуществляется с помощью агентов тестирования. Локальные агенты создаются и подключаются автоматически. Количество локальных агентов по умолчанию соответствует количеству ядер на компьютере.

У каждого агента тестирования своя копия глобальных переменных, которая никак не связана с клиентским терминалом. Сам терминал является диспетчером, который раздает задачи локальным и удаленным агентам. После выполнения очередного задания по тестированию советника с заданными параметрами агент возвращает терминалу результаты. При одиночном тестировании используется только один агент.

Агент хранит полученную от терминала историю в отдельных папках по имени инструмента, то есть история для EURUSD хранится в папке с именем EURUSD. Кроме того история инструментов разделяется по источникам. Структура для хранения истории выглядит таким образом:

```
каталог_тестера\Agent-IPaddress-Port\bases\имя_источника\history\имя_инструмента
```

Например, история по EURUSD с сервера MetaQuotes-Demo может храниться в папке каталог_тестера\Agent-127.0.0.1-3000\bases\MetaQuotes-Demo\EURUSD.

Локальный агент после окончания тестирования находится в режиме ожидания следующей задачи в течение 5 минут, чтобы не терять время на запуск при следующих вызовах. Только по истечении ожидания локальный агент прекращает свою работу и выгружается из памяти компьютера.

При досрочном завершении тестирования со стороны пользователя (кнопка "Отмена"), а также при закрытии клиентского терминала все локальные агенты тут же прекращают свою работу и выгружаются из памяти.

Обмен данными между терминалом и агентом

При запуске тестирования терминал готовит для отправки агенту несколько блоков параметров:

- Входные параметры тестирования (режим моделирования, интервал тестирования, инструмент, критерий оптимизации и т.д.)
- Список выбранных в "Обзоре рынка" инструментов
- Спецификация тестируемого инструмента (размер контракта, допустимые отступы от рынка для установки StopLoss и Takeprofit, и т.д)
- Тестируемый эксперт и значения его входных параметров
- Информация о дополнительных файлах (библиотеки, индикаторы, файлы данных - #property tester_...)

© 2000-2025, MetaQuotes Ltd.

tester_indicator	string	Имя пользовательского индикатора в формате "имя_индикатора.ex5". Необходимые для тестирования индикаторы определяются автоматически из вызова функций iCustom(), если соответствующий параметр задан константной строкой. Для остальных случаев (использование функции IndicatorCreate() или использование неконстантной строки в параметре, задающем имя индикатора) необходимо данное свойство
tester_file	string	Имя файла для тестера с указанием расширения, заключенное в двойные кавычки (как константная строка). Указанный файл будет передан тестеру в работу. Входные файлы для тестирования, если необходимы, должны указываться всегда
tester_library	string	Имя библиотеки с расширением, заключенное в двойные кавычки. Библиотека может быть как с расширением dll, так и с расширением ex5. Необходимые для тестирования библиотеки определяются автоматически. Однако, если какая-либо библиотека используется пользовательским индикатором, то необходимо использовать данное свойство

Для каждого блока параметров создается цифровой отпечаток в виде MD5-хэша, который и посылается агенту. MD5-хэш является уникальным для каждого набора, его объем во много раз меньше объема информации, на основе которой он вычислен.

Агент получает хэши блоков и с сравнивает с теми, что он уже хранит у себя. Если отпечаток данного блока параметров отсутствует у агента, или присланный хэш отличается от имеющегося, то агент запрашивает сам блок параметров. Таким образом уменьшается трафик между терминалом и агентом.

После проведения тестирования агент возвращает терминалу все результаты прогона, показываемые во вкладках "Результаты тестирования" и "Результаты оптимизации": полученная прибыль, количество сделок, коэффициент Шарпа, результат функции OnTester() и т.д.

При оптимизации терминал раздает агентам задачи на проведение тестирования небольшими пакетами, в каждом пакете находится несколько заданий (каждое задание означает одиночное тестирование с набором входных параметров). Это уменьшает время обмена между терминалом и агентом.

Агенты никогда не записывают на жесткий диск полученные от терминала EX5-файлы (эксперт, индикаторы, библиотеки и т.д.) из соображений безопасности, чтобы на компьютере с установленным агентом нельзя было воспользоваться присланными данным. Все остальные файлы, в том числе DLL, записываются в "песочницу". В удалённых агентах нельзя тестировать экспертов с использованием DLL.

Результаты тестирования складываются терминалом в специальный кэш результатов (результирующий кэш) для последующего быстрого доступа к ним при необходимости. Для каждого набора параметров терминал ищет в результирующем кэше уже готовые результаты от предыдущих запусков для исключения повторных запусков. Если результат с таким набором параметров не найден, агенту отдается задание на проведение тестирования.

Весь трафик между терминалом и агентами шифруется.

© 2000-2025, MetaQuotes Ltd.

Тики не пересылаются по сети, они генерируются на тестерных агентах.

Использование общей папки всех клиентских терминалов

Все тестерные агенты изолированы друг от друга и от клиентского терминала: у каждого агента есть собственная папка, в которую записываются логи агента. Кроме того, все файловые операции при тестирования агента происходят в папке **имя_агента/MQL5/Files**. Однако можно реализовать взаимодействие между локальными агентами и клиентским терминалом через общую папку всех клиентских терминалов, если при открытии файла указать флаг FILE_COMMON:

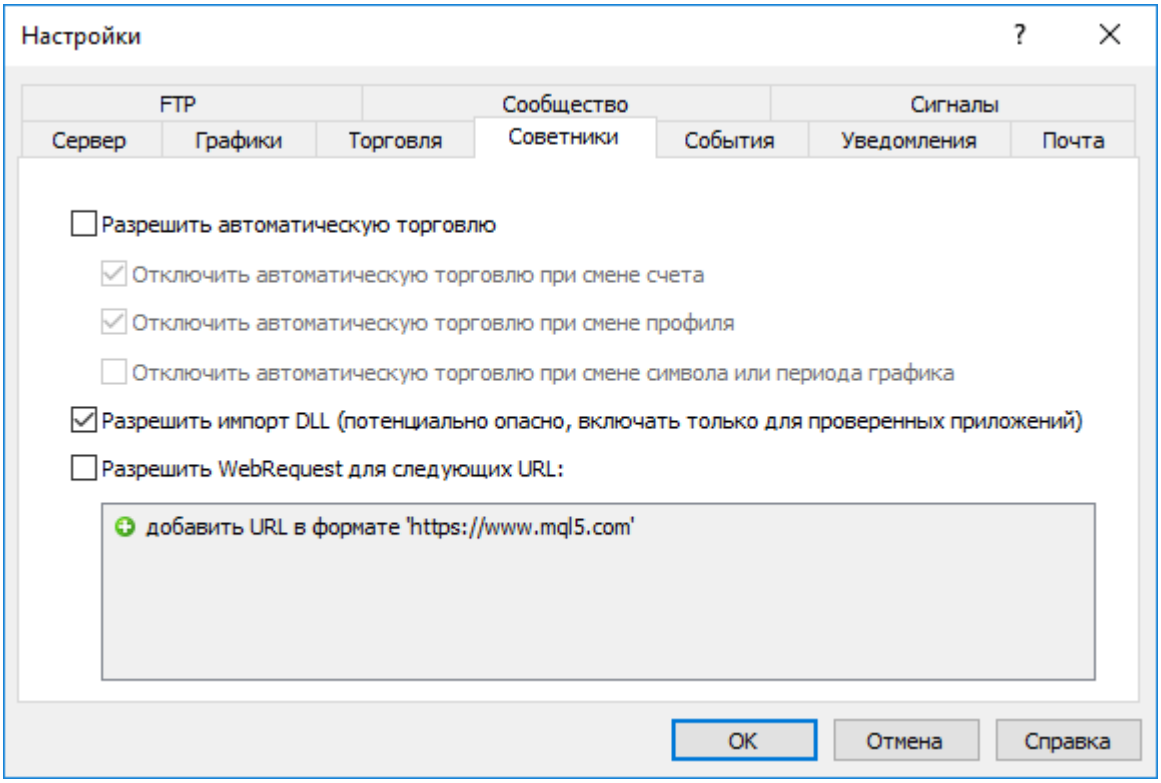
```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- общая папка всех клиентских терминалов
   common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- выведем имя этой папки
   PrintFormat("Откроем файл в общей папке клиентских терминалов %s", common_folder);
//--- откроем файл в общей папке (указан флаг FILE_COMMON)
   handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
   ... дальнейшие действия
//---
   return(INIT_SUCCEEDED);
  }
```

Использование DLL

Для ускорения оптимизации можно использовать не только локальные, но и удаленные агенты. При этом есть некоторые ограничения для удаленных агентов. Во-первых, удаленные агенты не выводят в свои логи результаты выполнения функции Print(), сообщения об открытии/закрытии позиций. Выводится в лог минимум информации чтобы неправильно написанные эксперты не забили сообщениями жесткий диск компьютера, на котором работает удаленный агент.

Второе ограничение - запрет на использование DLL при тестировании экспертов. Вызовы DLL безусловно запрещены на удалённых агентах из соображений безопасности. На локальных агентах вызовы dll в тестируемых экспертах разрешены только в том случае, если установлено соответствующее разрешение "Разрешить импорт DLL".

© 2000-2025, MetaQuotes Ltd.



Важно: при использовании полученных со стороны советников (скриптов, индикаторов), которые требуют разрешить вызовы DLL, вы должны осознавать весь риск, который принимаете на себя в случае разрешения этой опции в настройках терминала. Независимо от того, как будет использован советник - для тестирования или для запуска на графике.

© 2000-2025, MetaQuotes Ltd.

Предопределенные переменные

Для каждой выполняющейся mql5-программы поддерживается ряд предопределенных переменных, которые отражают состояние текущего ценового графика на момент запуска программы - эксперта, скрипта или пользовательского индикатора.

Значение предопределенным переменным устанавливает клиентский терминал перед запуском mql5-программы на выполнение. Предопределенные переменные константны и не могут быть изменены из mql5-программы. Исключение составляет переменная _LastError, которая может быть обнулена функцией ResetLastError.

Переменная	Значение
_AppliedTo	Переменная _AppliedTo позволяет узнать в индикаторе тип данных, на которых он рассчитывается
_Digits	Количество десятичных знаков после запятой
_Point	Размер пункта текущего инструмента в валюте котировки
_LastError	Значение последней ошибки
_Period	Значение таймфрейма текущего графика
_RandomSeed	Текущее состояние генератора псевдослучайных целых чисел
_StopFlag	Флаг остановки программы
_Symbol	Имя символа текущего графика
_UninitReason	Код причины деинициализации программы
_IsX64	Переменная _IsX64 позволяет узнать в каком терминале запущена MQL5-программа

Предопределенные переменные не могут быть определены в библиотеке. Библиотека использует эти переменные, определенные в программе, из которой эта библиотека была вызвана.

© 2000-2025, MetaQuotes Ltd.

int _AppliedTo

Переменная _AppliedTo позволяет узнать в индикаторе тип данных, на которых он рассчитывается:

Тип данных	Значение	Описание данных, используемых для расчета индикатора
—	0	Индикатор использует вторую форму вызова OnCalculate() - данные для расчета не задаются одним каким-то буфером или массивом данных
Close	1	Цены Close
Open	2	Цены Open
High	3	Цены High
Low	4	Цены Low
Median Price (HL/2)	5	Средняя цена = (High+Low)/2
Typical Price (HLC/3)	6	Типическая цена = (High+Low+Close)/3
Weighted Price (HLCC/4)	7	Взвешеннаяя цена = (Open+High+Low+Close)/4
Previous Indicator's Data	8	Данные индикатора, который был запущен на график перед данным индикатором
First Indicator's Data	9	Данные индикатора, который был запущен на график самым первым
Indicator handle	10+	Данные с индикатора, переданного в функцию iCustom() с помощью хендла индикатора. Значение _AppliedTo содержит хендл индикатора

Пример:

```
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
// получим тип данных, на котором считается индикатор
   Print("_AppliedTo=",_AppliedTo);
   Print(getIndicatorDataDescription(_AppliedTo));
//---
   return(INIT_SUCCEEDED);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Описание данных, на которых расчитывается индикатор              |
//+------------------------------------------------------------------+
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(d
         break;
     }
//---
   return descr;
  }
```

Смотри также

ENUM_APPLIED_PRICE

© 2000-2025, MetaQuotes Ltd.

int _Digits

В переменной _Digits хранится количество десятичных знаков после запятой, определяющее точность измерения цены символа текущего графика.

Можно также использовать функцию Digits().

© 2000-2025, MetaQuotes Ltd.

double _Point

В переменной _Point хранится размер пункта текущего инструмента в валюте котировки.

Можно также использовать функцию Point().

© 2000-2025, MetaQuotes Ltd.

int _LastError

В переменной _LastError хранится значение последней ошибки, произошедшей во время исполнения mql5-программы. Сбросить значение в ноль можно функцией ResetLastError().

Для получения кода ошибки можно также использовать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

ENUM_TIMEFRAMES _Period

В переменной _Period хранится значение таймфрейма текущего графика.

Можно также использовать функцию Period().

Смотри также

PeriodSeconds, Периоды графиков, Дата и время, Видимость объектов

© 2000-2025, MetaQuotes Ltd.

_RandomSeed

Переменная для хранения текущего состояния при генерации псевдослучайных целых чисел. _RandomSeed меняет своё значение при вызове MathRand(). Для установки нужного начального состояния используйте MathSrand().

Случайное число ***x***, получаемое функцией MathRand(), вычисляется при каждом вызове следующим образом:

```
x=_RandomSeed*214013+2531011;
_RandomSeed=x;
x=(x>>16)&0x7FFF;
```

Смотри также

MathRand(), MathSrand(), Целые типы

© 2000-2025, MetaQuotes Ltd.

int _StopFlag

В переменной _StopFlag хранится флаг остановки mql5-программы, который в обычном режиме работы равен 0. Когда клиентский терминал пытается остановить программу, в эту переменную записывается значение, отличное от нуля.

Для проверки значения флага _StopFlag можно также использовать функцию IsStopped().

© 2000-2025, MetaQuotes Ltd.

string _Symbol

В переменной _Symbol хранится имя символа текущего графика.

Можно также использовать функцию Symbol().

© 2000-2025, MetaQuotes Ltd.

int _UninitReason

В переменной _UninitReason хранится код причины деинициализации программы.

Обычно код причины деинициализации получают с помощью функции UninitializeReason().

© 2000-2025, MetaQuotes Ltd.

int _IsX64

Переменная _IsX64 позволяет узнать в каком терминале запущена MQL5-программа: для 32-х битного терминала _IsX64=0 и для 64-х битного терминала _IsX64!=0.

Можно также использовать функцию TerminalInfoInteger(TERMINAL_X64).

Пример:

```
// проверим в каком терминале запущена программа
   Print("_IsX64=",_IsX64);
   if(_IsX64)
      Print("Программа ",__FILE__," запущена в 64-битном терминале");
   else
      Print("Программа ",__FILE__," запущена в 32-битном терминале");
   Print("TerminalInfoInteger(TERMINAL_X64)=",TerminalInfoInteger(TERMINAL_X64));
```

Смотри также

MQLInfoInteger, Импорт функций (#import)

© 2000-2025, MetaQuotes Ltd.

Общие функции

Функции общего назначения, которые не вошли ни в одну из специализированных групп.

Функция	Действие
Alert	Выводит сообщение в отдельном окне
CheckPointer	Возвращает тип указателя объекта
Comment	Выводит сообщение в левый верхний угол ценового графика
CryptEncode	Преобразует данные массива-источника в массив-приемник указанным методом
CryptDecode	Производит обратное преобразование данных массива
DebugBreak	Программная точка останова при отладке
ExpertRemove	Прекращает работу эксперта и выгружает его с графика
GetPointer	Возвращает указатель объекта
GetTickCount	Возвращает количество миллисекунд, прошедших с момента старта системы
GetTickCount64	Возвращает количество миллисекунд, прошедших с момента старта системы
GetMicrosecondCount	Возвращает количество микросекунд, прошедших с момента начала работы MQL5-программы
MessageBox	Создает и отображает окно сообщений, а также управляет им
PeriodSeconds	Возвращает количество секунд в периоде
PlaySound	Воспроизводит звуковой файл
Print	Выводит сообщение в журнал
PrintFormat	Форматирует и печатает наборы символов и значений в лог-файл в соответствие с заданным форматом
ResetLastError	Устанавливает значение предопределенной переменной _LastError в ноль
ResourceCreate	Создает ресурс изображения на основе набора данных
ResourceFree	Удаляет динамически созданный ресурс (освобождает занятую ресурсом память)
ResourceReadImage	Читает данные графического ресурса, созданного функцией ResourceCreate() или сохраненного в EX5-файле при компиляции
ResourceSave	Сохраняет ресурс в указанный файл
SetUserError	Устанавливает предопределенную переменную _LastError в значение, равное ERR_USER_ERROR_FIRST + user_error

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
SetReturnError	Устанавливает код возврата, который вернёт процесс терминала при завершение работы
Sleep	Задерживает выполнение текущего эксперта или скрипта на определенный интервал
TerminalClose	Посылает терминалу команду на завершение работы
TesterHideIndicators	Задает режим показа/сокрытия индикаторов, которые используются в эксперте
TesterStatistics	Возвращает значение указанного статистического показателя, рассчитанного по результатам тестирования
TesterStop	Отдает команду на завершении работы программы при тестировании.
TesterDeposit	Эмуляции операций внесения средств в процессе тестирования. Может быть использована в некоторых системах управления капиталом.
TesterWithdrawal	Эмуляция операций снятия средств в процессе тестирования
TranslateKey	Возвращает Unicode-символ по виртуальному коду клавиши
ZeroMemory	Обнуляет переменную, переданную по ссылке. Тип переменной может быть любым, исключение составляют только классы и структуры, имеющие конструкторы

© 2000-2025, MetaQuotes Ltd.

Alert

Отображает диалоговое окно, содержащее пользовательские данные.

```
void  Alert(
   argument,      // первое значение
   ...            // последующие значения
   );
```

Параметры

argument

[in] Любые значения, разделенные запятыми. Для разделения выводимой информации на несколько строк можно использовать символ перевода строки "\n" либо "\r\n". Количество параметров не может превышать 64.

Возвращаемое значение

Нет возвращаемого значения

Примечание

Массивы нельзя передавать в функцию Alert(). Массивы должны выводиться поэлементно. Данные типа double выводятся с 8 десятичными цифрами после точки, данные типа float выводятся с 5 десятичными цифрами после точки. Для вывода вещественных чисел с другой точностью либо в научном формате необходимо использовать функцию DoubleToString().

Данные типа bool выводятся в виде строк "true" или "false". Даты выводятся в виде YYYY.MM.DD HH:MI:SS. Для вывода даты в другом формате необходимо использовать функцию TimeToString(). Данные типа color выводятся либо в виде строки R,G,B, либо в виде названия цвета, если этот цвет присутствует в наборе цветов.

При работе в тестере стратегий функция Alert() не выполняется.

Пример:

```
//--- enums
enum ENUM_INTERSECT_DIRECTION
  {
   INTERSECT_DIRECTION_NONE= 0,  // нет пересечения
   INTERSECT_DIRECTION_UP  = 1,  // пересечение вверх
   INTERSECT_DIRECTION_DOWN=-1,  // пересечение вниз
  };

//--- input parameters
input    uint               InpPeriod = 10;            // MA Period
input    int                InpShift  = 0;             // MA Shift
input    ENUM_MA_METHOD     InpMethod = MODE_SMA;      // MA Method
input    ENUM_APPLIED_PRICE InpPrice  = PRICE_CLOSE;   // MA Applied price

//--- global variables
```

© 2000-2025, MetaQuotes Ltd.

```
int      ExtMaHandle;
int      ExtMaPeriod;
double   ExtData[2];
MqlRates ExtRates[2];

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- период расчёта скользящей средней будет равен значению по умолчанию (10) в случа
   ExtMaPeriod=int(InpPeriod<1 ? 10 : InpPeriod);
//--- создаём хэндл индикатора Moving Average с указанными параметрами
   ExtMaHandle=iMA(Symbol(),PERIOD_CURRENT,ExtMaPeriod,InpShift,InpMethod,InpPrice);
   ResetLastError();
   if(ExtMaHandle==INVALID_HANDLE)
     {
      PrintFormat("Failed to create iMA() handle. Error code: %d",GetLastError());
      return(INIT_FAILED);
     }

//--- получаем время последнего обновления цен
   datetime tick_time=TickTime();
//--- получаем данные скользящей средней и данные цен с двух последних баров
   if(GetData(ExtMaHandle,ExtData,ExtRates) && tick_time!=0)
     {
      //--- если цена выше скользящей средней
      if(ExtRates[1].close>ExtData[1])
        {
         //--- создаём текст сообщения и выводим Alert
         string message=StringFormat("Bar time: %s. The price is above the moving aver
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SECON
         /*
         Результат:
         Alert: Bar time: 2024.02.16 18:00. The price is above the moving average at 2
         */
        }
      else
        {
         //--- если цена ниже скользящей средней
         if(ExtRates[1].close<ExtData[1])
           {
            //--- создаём текст сообщения и выводим Alert
            string message=StringFormat("Bar time: %s. The price is below the moving a
            Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SE
            /*
            Результат:
            Alert: Bar time: 2024.02.16 19:00. The price is below the moving average a
            */
```

© 2000-2025, MetaQuotes Ltd.

```
            }
          else
            {
             //--- создаём текст сообщения и выводим Alert
             string message=StringFormat("Bar time: %s. The price and moving average ar
             Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SE
             /*
             Результат:
             Alert: Bar time: 2024.02.16 20:00. The price and moving average are equal
             */
            }
         }
      }

//--- Успешно
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   ResetLastError();
//--- получаем данные скользящей средней и данные цен с двух последних баров
   if(!GetData(ExtMaHandle,ExtData,ExtRates))
      return;
//--- получаем направление пересечения ценой скользящей средней на текущем баре
   ENUM_INTERSECT_DIRECTION intersect=GetIntersectDirection(ExtData,ExtRates);

//--- переменная для сохранения предыдущего сообщения
   static string message_prev="";

//--- если цена пересекла скользящую среднюю на текущем баре снизу-вверх
   if(intersect==INTERSECT_DIRECTION_UP)
     {
      //--- получаем время тика, на котором произошло пересечение
      datetime tick_time=TickTime();
      if(tick_time==0)
         return;
      //--- создаём текст сообщения
      string message=StringFormat("Bar time: %s. The price crossed the MA from bottom
      //--- если прошлое сообщение не равно текущему - выводим Alert с сообщением и вр
      if(message!=message_prev)
        {
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SECON
         message_prev=message;
         /*
         Результат:\
         Alert: Bar time: 2024.02.16 09:00. The price crossed the MA from bottom to to
```

© 2000-2025, MetaQuotes Ltd.

```
         */
        }
     }

//--- если цена пересекла скользящую среднюю на текущем баре сверху-вниз
   if(intersect==INTERSECT_DIRECTION_DOWN)
     {
      //--- получаем время тика, на котором произошло пересечение
      datetime tick_time=TickTime();
      if(tick_time==0)
         return;
      //--- создаём текст сообщения
      string message=StringFormat("Bar time: %s. The price crossed the MA from top to
      //--- если прошлое сообщение не равно текущему - выводим Alert с сообщением и вр
      if(message!=message_prev)
        {
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SECON
         message_prev=message;
         /*
         Результат:\
         Alert: Bar time: 2024.02.16 10:00. The price crossed the MA from top to botto
         */
        }
     }
  }
//+------------------------------------------------------------------+
//| Получает данные цены и скользящей средней в массивы              |
//+------------------------------------------------------------------+
bool GetData(int handle,double &ma_data[],MqlRates &price_data[])
  {
   ResetLastError();
//--- получаем данные скользящей средней с двух последних баров
   if(CopyBuffer(handle,0,0,2,ma_data)!=2)
     {
      PrintFormat("CopyBuffer() failed. Error code: %d",GetLastError());
      return(false);
     }
//--- получаем данные цен двух последних баров
   if(CopyRates(Symbol(),PERIOD_CURRENT,0,2,price_data)!=2)
     {
      PrintFormat("CopyRates() failed. Error code: %d",GetLastError());
      return(false);
     }

   return(true);
  }
//+------------------------------------------------------------------+
//| Возвращает направление пересечения ценой скользящей средней     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
ENUM_INTERSECT_DIRECTION GetIntersectDirection(double &ma_data[],MqlRates &price_data
  {
   double ma0=ma_data[1];
   double ma1=ma_data[0];
   double close0=price_data[1].close;
   double close1=price_data[0].close;

   if(close1<=ma1 && close0>ma0)
      return(INTERSECT_DIRECTION_UP);
   else
     {
      if(close1>=ma1 && close0<ma0)
         return(INTERSECT_DIRECTION_DOWN);
      else
         return(INTERSECT_DIRECTION_NONE);
     }
  }
//+------------------------------------------------------------------+
//| Возвращает время тика в секундах                                 |
//+------------------------------------------------------------------+
datetime TickTime()
  {
   MqlTick tick={};

   ResetLastError();
   if(!SymbolInfoTick(Symbol(),tick))
     {
      PrintFormat("SymbolInfoTick() failed. Error code: %d",GetLastError());
      return(0);
     }

   return(tick.time);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckPointer

Возвращает тип указателя объекта.

```
ENUM_POINTER_TYPE  CheckPointer(
   object* anyobject      // указатель объекта
   );
```

Параметры

anyobject

[in] Указатель объекта.

Возвращаемое значение

Возвращает значение из перечисления ENUM_POINTER_TYPE.

Примечание

Попытка обращения к некорректному указателю приводит к критическому завершению программы. Поэтому существует необходимость использования функции CheckPointer перед использованием указателя. Указатель может быть некорректным в следующих случаях:

- указатель равен NULL;
- если объект был уничтожен при помощи оператора delete.

Данную функцию можно использовать как проверку указателя на корректность. Значение, отличное от нуля, гарантирует, что по этому указателю можно получить доступ к данным.

Для быстрой проверки указателя можно также использовать оператор "!"(пример), который проверяет его на валидность через неявный вызов функции CheckPointer.

Пример:

```
//+------------------------------------------------------------------+
//| Уничтожает список через уничтожение элементов                    |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- служебный указатель для работы в цикле
   CItem* item;
//--- пройдемся в цикле и попытаемся удалить динамические указатели
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dynamyc object ",item.Identifier()," to be deleted");
         delete (item);
        }
      else
         Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//---
  }
```

Смотри также

Указатели объектов, Проверка указателя объекта, Оператор уничтожения объекта delete

© 2000-2025, MetaQuotes Ltd.

Comment

Выводит комментарий, определенный пользователем, в левый верхний угол графика.

```
void  Comment(
   argument,      // первое значение
   ...            // последующие значения
   );
```

Параметры

...

[in] Любые значения, разделенные запятыми. Для разделения выводимой информации на несколько строк можно использовать символ перевода строки "\n" либо "\r\n". Количество параметров не может превышать 64. Общая длина выводимого сообщения (включая служебные неотображаемые символы) не может превышать 2045 символов (лишние символы будут обрезаны при выводе).

Возвращаемое значение

Нет возвращаемого значения

Примечание

Массивы нельзя передавать в функцию Comment(). Массивы должны печататься поэлементно.

Данные типа double выводятся с точностью до 16 десятичных цифр после точки, при этом данные могут выводиться либо в традиционном либо в научном формате - в зависимости от того, как запись будет наиболее компактна. Данные типа float выводятся с 5 десятичными цифрами после точки. Для вывода вещественных чисел с другой точностью либо в явно указанном формате необходимо использовать функцию DoubleToString().

Данные типа bool выводятся в виде строк "true" или "false". Даты выводятся в виде YYYY.MM.DD HH:MI:SS. Для вывода даты в другом формате необходимо использовать функцию TimeToString(). Данные типа color выводятся либо в виде строки R,G,B, либо в виде названия цвета, если этот цвет присутствует в наборе цветов.

При работе в тестере стратегий в режиме оптимизации функция Comment() не выполняется.

Пример:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
//--- Выведем значения в три строчки
   Comment(StringFormat("Выводим цены\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,Spread
  }
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

ChartSetString, ChartGetString

© 2000-2025, MetaQuotes Ltd.

CryptEncode

Преобразует данные массива-источника в массив-приемник указанным методом.

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,         // метод преобразования
   const uchar&        data[],         // массив-источник
   const uchar&        key[],          // ключ шифрования
   uchar&              result[]        // массив-приемник
   );
```

Параметры

method

[in] Метод преобразования. Может быть одним из значений перечисления ENUM_CRYPT_METHOD.

data[]

[in] Массив-источник.

key[]

[in] Ключ шифрования.

result[]

[out] Массив-приемник.

Возвращаемое значение

Количество байт в массиве-приемнике или 0 в случае ошибки. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";
//--- проверка размера
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);
//--- преобразование в шестнадцатиричную строку
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   string text="The quick brown fox jumps over the lazy dog";
   string keystr="ABCDEFG";
   uchar src[],dst[],key[];
//--- подготовка ключа шифрования
   StringToCharArray(keystr,key);
//--- подготовка исходного массива src[]
   StringToCharArray(text,src);
//--- вывод исходных данных
   PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(s
//--- шифрование массива src[] методом DES с 56-битным ключом key[]
   int res=CryptEncode(CRYPT_DES,src,key,dst);
//--- проверка результата шифрования
   if(res>0)
     {
      //--- вывод шифрованных данных
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- расшифровка данных массива dst[] методом DES с 56-битным ключом key[]
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- проверка результата
      if(res>0)
        {
         //--- вывод дешифрованных данных
         PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("Ошибка в CryptDecode. Код ошибки=",GetLastError());
     }
   else
      Print("Ошибка в CryptEncode. Код ошибки=",GetLastError());
  }
```

Смотри также

Операции с массивами, CryptDecode()

© 2000-2025, MetaQuotes Ltd.

CryptDecode

Производит обратное преобразование данных массива, полученного при помощи функции CryptEncode().

```
int  CryptDecode(
   ENUM_CRYPT_METHOD   method,        // метод преобразования
   const uchar&        data[],        // массив-источник
   const uchar&        key[],         // ключ шифрования
   uchar&              result[]       // массив-приемник
   );
```

Параметры

method

[in] Метод преобразования. Может быть одним из значений перечисления ENUM_CRYPT_METHOD.

data[]

[in] Массив-источник.

key[]

[in] Ключ шифрования.

result[]

[out] Массив-приемник.

Возвращаемое значение

Количество байт в массиве-приемнике или 0 в случае ошибки. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
input string InpKey = "ABCDEFG";  // Encryption key

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string text="The quick brown fox jumps over the lazy dog";
   uchar src[],dst[],key[];

//--- подготовка ключа шифрования
   StringToCharArray(InpKey,key);
//--- подготовка исходного массива src[]
   StringToCharArray(text,src);
//--- вывод исходных данных
   PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(s
```

© 2000-2025, MetaQuotes Ltd.

```
//--- шифрование массива src[] методом DES с 56-битным ключом key[]
   int res=CryptEncode(CRYPT_DES,src,key,dst);

//--- проверка результата шифрования
   if(res>0)
     {
      //--- вывод шифрованных данных
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- расшифровка данных массива dst[] методом DES с 56-битным ключом key[]
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- проверка результата
      if(res>0)
        {
         //--- вывод дешифрованных данных
         PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("CryptDecode failed. Error: ",GetLastError());
     }
   else
      Print("CryptEncode failed. Error: ",GetLastError());
  }
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";

//--- проверка размера
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);

//--- преобразование в шестнадцатеричную строку
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);

   return(res);
  }
```

Смотри также

Операции с массивами, CryptEncode()

© 2000-2025, MetaQuotes Ltd.

DebugBreak

Программная точка останова при отладке.

```
void  DebugBreak();
```

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Прерывание выполнения mql5-программы происходит только в том случае, если программа запущена в режиме отладки. Функцию можно использовать для просмотра значений переменных и/или дальнейшего пошагового выполнения.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- скомпилировать файл по F5
//--- в режиме отладки при i == j остановимся на строке DebugBreak()
   for(int i=0,j=20; i<20; i++,j--)
     {
      if(i==j)
         DebugBreak();
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

ExpertRemove

Прекращает работу эксперта и выгружает его с графика.

```
void  ExpertRemove();
```

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Остановка эксперта не происходит немедленно при вызове функции ExpertRemove(), производится лишь взвод флага для прекращения работы эксперта. Т.е., любое следующее событие эксперт обрабатывать уже не будет, произойдет вызов OnDeinit() и выгрузка с удалением с графика.

Вызов ExpertRemove() в тестере стратегий внутри обработчика OnInit() приведет к отмене тестирования на текущем наборе параметров. Такое завершение рассматривается как ошибка при инициализации.

При вызове ExpertRemove() в тестере стратегий после успешной инициализации советника тестирование завершится штатным образом с вызовом OnDeinit() и OnTester(). В этом случае будет получена вся торговая статистика и значение критерия оптимизации.

Пример:

```
//+------------------------------------------------------------------+
//|                                            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;// количество тиков до снятия эксперта
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(TimeCurrent(),": " ,__FUNCTION__," reason code = ",reason);
//--- "clear" comment
   Comment("");
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   static int tick_counter=0;
//---
   tick_counter++;
   Comment("\nДо выгрузки эксперта ",__FILE__," осталось ",
           (ticks_to_close-tick_counter)," тиков ");
//--- до
   if(tick_counter>=ticks_to_close)
     {
      ExpertRemove();
      Print(TimeCurrent(),": ",__FUNCTION__," эксперт будет выгружен");
     }
   Print("tick_counter = ",tick_counter);
//---
  }
//+------------------------------------------------------------------+
```

Смотри также

Выполнение программ, События клиентского терминала

© 2000-2025, MetaQuotes Ltd.

GetPointer

Возвращает указатель объекта.

```
void*  GetPointer(
   any_class anyobject      // объект любого класса
   );
```

Параметры

anyobject

[in] Объект любого класса.

Возвращаемое значение

Возвращает указатель объекта.

Примечание

Только объекты классов имеют указатели. Экземпляры структур и переменные простых типов указателей не имеют. Объект класса, который не создан при помощи оператора new(), а например, автоматически созданный в массиве объектов, все равно имеет указатель. Только этот указатель будет иметь автоматический тип POINTER_AUTOMATIC, и к нему нельзя применить оператор delete(). В остальном указатель типа ничем не отличается от динамических указателей типа POINTER_DYNAMIC.

Так как переменные типа структур и простых типов не имеют указателей, то применять к ним функцию GetPointer() запрещено. Также запрещается в качества аргумента функции передавать указатель. Во всех перечисленных случаях компилятор сообщит об ошибке.

Попытка к обращения к некорректному указателю приводит к критическому завершению программы. Поэтому существует необходимость использования функции CheckPointer() перед использованием указателя. Указатель может быть некорректным в следующих случаях:

- указатель равен NULL;
- если объект был уничтожен при помощи оператора delete.

Данную функцию можно использовать как проверку указателя на корректность. Значение, отличное от нуля, гарантирует, что по этому указателю можно получить доступ к данным.

Пример:

```
//+------------------------------------------------------------------+
//|                                              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"

//+------------------------------------------------------------------+
//| Класс, реализующий элемент списка                                |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
class CItem
  {
   int               m_id;
   string            m_comment;
   CItem*            m_next;
public:
                     CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
                    ~CItem() { Print("Destructor of ",m_id,
                                     (CheckPointer(GetPointer(this))==POINTER_DYNAMIC)
                                     "dynamic":"non-dynamic"); }
   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; }
  };
//+------------------------------------------------------------------+
//| Простейший класс списка                                          |
//+------------------------------------------------------------------+
class CMyList
  {
   CItem*            m_items;
public:
                     CMyList() { m_items=NULL; }
                    ~CMyList() { Destroy(); }
    bool             InsertToBegin(CItem* item);
    void             Destroy();
  };
//+------------------------------------------------------------------+
//| Вставляет элемент списка в самое начало                          |
//+------------------------------------------------------------------+
bool CMyList::InsertToBegin(CItem* item)
  {
   if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
   item.Next(m_items);
   m_items=item;
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Уничтожает список через уничтожение элементов                    |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- служебный указатель для работы в цикле
   CItem* item;
//--- пройдемся в цикле и попытаемся удалить динамические указатели
   while(CheckPointer(m_items)!=POINTER_INVALID)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       item=m_items;
       m_items=m_items.Next();
       if(CheckPointer(item)==POINTER_DYNAMIC)
         {
          Print("Dynamyc object",item.Identifier(),"to be deleted");
          delete (item);
         }
       else Print("Non-dynamic object",item.Identifier(),"cannot be deleted");
      }
//---
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   CMyList list;
   CItem   items[10];
   CItem*  item;
//--- создадим и добавим в список динамический указатель объекта
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(100,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- добавим автоматические указатели в список
   for(int i=0; i<10; i++)
     {
      items[i].Initialize(i,"automatic");
      items[i].PrintMe();
      item=GetPointer(items[i]);
      if(CheckPointer(item)!=POINTER_INVALID)
         list.InsertToBegin(item);
     }
//--- добавим еще один динамический указатель объекта в начало списка
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(200,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- удалим элементы списка
   list.Destroy();
//--- все элементы списка будут уничтожены,
//--- смотри в терминале закладку Experts
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

Смотри также

Указатели объектов, Проверка указателя объекта, Оператор уничтожения объекта delete

© 2000-2025, MetaQuotes Ltd.

GetTickCount

Функция GetTickCount() возвращает количество миллисекунд, прошедших с момента старта системы.

```
uint  GetTickCount();
```

Возвращаемое значение

Значение типа uint.

Примечание

Счетчик ограничен разрешающей способностью системного таймера. Так как время хранится как беззнаковое целое, то он переполняется каждые 49.7 дней при непрерывной работе компьютера.

Пример:

```
#define MAX_SIZE 40
//+------------------------------------------------------------------+
//| скрипт для замера времени вычисления 40 чисел Фибоначчи          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- запомним начальное значение
   uint start=GetTickCount();
//--- переменная для получения очередного числа из ряда Фибоначчи
   long fib=0;
//--- цикл, в котором вычисляем заданное количество чисел из ряда Фибоначчи
   for(int i=0;i<MAX_SIZE;i++)
      fib=TestFibo(i);
//--- получим затраченное время в миллисекундах
   uint time=GetTickCount()-start;
//--- выведем в журнал "Эксперты" сообщение
   PrintFormat("Вычисление %d первых чисел Фибоначчи заняло %d ms",MAX_SIZE,time);
//--- работа скрипта завершена
   return;
  }
//+------------------------------------------------------------------+
//| Функция получения числа Фибоначчи по его порядковому номеру      |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- первый член ряда Фибоначчи
   if(n<2) return(1);
//--- все последующие члены вычисляются по этой формуле
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

Дата и время, EventSetMillisecondTimer, GetTickCount64, GetMicrosecondCount

© 2000-2025, MetaQuotes Ltd.

GetTickCount64

Функция GetTickCount64() возвращает количество миллисекунд, прошедших с момента старта системы.

```
ulong  GetTickCount64();
```

Возвращаемое значение

Значение типа ulong.

Примечание

Счетчик ограничен разрешающей способностью системного таймера, который как правило возвращает результат с точностью 10-16 миллисекунд. В отличие от функции GetTickCount, которая имеет тип uint и поэтому переполняется каждые 49.7 дней при непрерывной работе компьютера, GetTickCount64() может использоваться при неограниченном времени работы компьютера и не боится переполнения.

Пример:

```
#define MAX_SIZE 40

//+------------------------------------------------------------------+
//| Скрипт для замера времени вычисления 40 чисел Фибоначчи          |
//+------------------------------------------------------------------+
void OnStart()
  {
   long fib_array[MAX_SIZE];

//--- запомним начальное значение
   ulong start=GetTickCount64();
//--- цикл, в котором вычисляем заданное количество чисел из ряда Фибоначчи
   for(int i=0;i<MAX_SIZE;i++)
      fib_array[i]=TestFibo(i);
//--- получим затраченное время в миллисекундах
   ulong time=GetTickCount64()-start;

//--- выведем в журнал "Эксперты" сообщение
   ArrayPrint(fib_array);
   PrintFormat("Calculating the first %d Fibonacci numbers took %I64u ms",MAX_SIZE,tim
  }
//+------------------------------------------------------------------+
//| Функция получения числа Фибоначчи по его порядковому номеру      |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- первый член ряда Фибоначчи
   if(n<2)
      return(1);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- все последующие члены вычисляются по этой формуле
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

Смотри также

Дата и время, EventSetMillisecondTimer, GetTickCount, GetMicrosecondCount

© 2000-2025, MetaQuotes Ltd.

GetMicrosecondCount

Функция GetMicrosecondCount() возвращает количество микросекунд, прошедших с момента начала работы MQL5-программы.

```
ulong  GetMicrosecondCount();
```

Возвращаемое значение

Значение типа ulong.

Пример:

```
//+------------------------------------------------------------------+
//| Тестируемый код                                                  |
//+------------------------------------------------------------------+
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++;
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   uint   ui=0,ui_max=0,ui_min=INT_MAX;
   ulong  ul=0,ul_max=0,ul_min=INT_MAX;
//--- количество тестов
   for(int count=0;count<1000;count++)
     {
      uint  ui_res=0;
      ulong ul_res=0;
      //---
      for(int n=0;n<2;n++)
        {
         //--- выбираем способ измерения
         if(n==0)
            ui=GetTickCount();
         else
            ul=GetMicrosecondCount();
         //--- тестируемый код
```

© 2000-2025, MetaQuotes Ltd.

```
         Test();
         //--- копим результат измерения (в зависимости от способа)
         if(n==0)
            ui_res+=GetTickCount()-ui;
         else
            ul_res+=GetMicrosecondCount()-ul;
        }
      //--- собираем минимальное и максимальное время исполнения кода обоих измерений
      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));
  }
```

Смотри также

Дата и время, GetTickCount, GetTickCount64

© 2000-2025, MetaQuotes Ltd.

MessageBox

Создает и отображает окно сообщений, а также управляет им. Окно сообщений содержит сообщение и заголовок, любую комбинацию предопределенных значков и командных кнопок.

```
int  MessageBox(
   string  text,              // текст сообщения
   string  caption=NULL,      // заголовок окна
   int     flags=0            // определяет набор кнопок в окне
   );
```

Параметры

text

[in] Текст, содержащий сообщение для отображения.

caption=NULL

[in] Необязательный текст для отображения в заголовке окна сообщения. Если этот параметр пустой, в заголовке окна будет отображено название эксперта.

flags=0

[in] Необязательные флаги, определяющие вид и поведение диалогового окна. Флаги могут быть комбинацией специальной группы флагов .

Возвращаемое значение

Если функция успешно выполняется, возвращаемое значение - одно из значений кодов возврата MessageBox().

Примечание

Функция запрещена в пользовательских индикаторах, так как вызов MessageBox() приостанавливает работу потока исполнения на всё время ожидания ответа пользователя. А так как все индикаторы по каждому символу выполняются в едином потоке, то такая остановка сделает невозможной работу всех графиков на всех таймфреймах по данному символу.

При работе в тестере стратегий функция MessageBox() не выполняется.

Пример:

```
//+------------------------------------------------------------------+
//| Советник выводит окно MessageBox с запросом о дальнейшей работе  |
//| при достижении указанного  количества серии убыточных сделок     |
//| Ожидает заданное количество баров и выводит окно MessageBox      |
//| с запросом о продолжении работы.                                 |
//| Для проверки достаточно вручную открыть и закрыть с убытком      |
//| несколько позиций, так как для упрощения советник не контролирует|
//| "свои" позиции по магику.                                        |
//+------------------------------------------------------------------+

//--- input parameters
input uint InpMaxLossDeals      =  3;    // Max Loss deals
```

© 2000-2025, MetaQuotes Ltd.

```
input uint InpInactivityNumBars =  5;     // Number of bars of advisor inactivity

//--- global variables
bool     ExtFirstStart=true;              // Флаг первого запуска
bool     ExtFlag=true;                    // Флаг разрешения работы советника
uint     ExtNumLoss;                      // Количество убыточных сделок подряд
datetime ExtTimeLastLoss;                 // Время последней сделки закрытия убыточной

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- получаем количество убыточных сделок подряд и время последней сделки закрытия по
   ExtNumLoss=GetNumLosingTradesInRow(ExtTimeLastLoss);

   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   Comment("");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- определяем, сколько прошло баров после последней закрытой убыточной позиции в се
   int bars_remaining=iBarShift(Symbol(),PERIOD_CURRENT,ExtTimeLastLoss);

//--- если это первый запуск
   if(ExtFirstStart)
     {
      //--- Если уже прошло заданное количество баров после серии убыточных позиций -
      if(bars_remaining>(int)InpInactivityNumBars)
         ExtFlag=true;
      ExtFirstStart=false;
     }

//--- если флаг работы советника снят
   if(!ExtFlag)
     {
      Comment(StringFormat("The advisor is stopped for %d bars. Num Loss positions: %u
                           (InpInactivityNumBars-bars_remaining),ExtNumLoss,TimeToStrin
      //--- если прошло заданное количество баров после серии убыточных позиций
      if(bars_remaining>(int)InpInactivityNumBars)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- выводим окно MessageBox с заданным текстом и заголовком окна
          //--- окно запроса имеет две кнопки Yes/No и иконку со знаком вопроса.
          //--- кнопка Yes выбрана по умолчанию.
          string mb_text="The specified number of bars of EA inactivity have passed.\n
          string mb_caption="Please note";
          int    mb_id=MessageBox(mb_text,mb_caption,MB_YESNO|MB_ICONQUESTION|MB_DEFBUT
          //--- если код возврата от MessageBox - нажатая кнопка Yes - устанавливаем фл
          if(mb_id==IDYES)
            {
             ExtFlag=true;
             return;
            }
         }
       //--- флаг работы советника снят - выходим из OnTick()
       return;
      }

//---  флаг работы советника установлен - работает советник, как предусмотрено кодом,
   Comment(StringFormat("The advisor is working. Num Loss positions: %u, Time last los
           ExtNumLoss,TimeToString(ExtTimeLastLoss,TIME_DATE|TIME_MINUTES|TIME_SECONDS
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction& trans,
                        const MqlTradeRequest& request,
                        const MqlTradeResult& result)
  {
//--- если тип транзакции - добавление сделки в историю
   if(trans.type==TRADE_TRANSACTION_DEAL_ADD)
     {
      //--- Получаем тикет сделки и выбираем сделку из списка по тикету
      ulong deal_ticket=trans.deal;
      if(HistoryDealSelect(deal_ticket))
        {
         //--- если это сделка на выход из рынка - получаем количество убыточных сдело
         ENUM_DEAL_ENTRY entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(deal_ticket,DEAI
         if(entry==DEAL_ENTRY_OUT || entry==DEAL_ENTRY_INOUT || entry==DEAL_ENTRY_OUT_
            ExtNumLoss=GetNumLosingTradesInRow(ExtTimeLastLoss);
        }
     }

//--- если количество убыточных сделок, идущих подряд, больше заданного значения и уст
   if(ExtNumLoss>=InpMaxLossDeals && ExtFlag)
     {
      //--- выводим окно MessageBox с заданным текстом и заголовком окна
      //--- окно запроса имеет две кнопки Yes/No и иконку с восклицательным знаком.
      //--- кнопка No выбрана по умолчанию.
```

© 2000-2025, MetaQuotes Ltd.

```
      string mb_text="The number of losing trades has reached the specified maximum. T
      string mb_caption="Attention!";
      int    mb_id=MessageBox(mb_text,mb_caption,MB_YESNO|MB_ICONQUESTION|MB_DEFBUTTON
      //--- если код возврата от MessageBox - нажатая кнопка No - снимаем флаг работы
      if(mb_id==IDNO)
         ExtFlag=false;
     }
  }
//+------------------------------------------------------------------+
//| Возвращает количество убыточных сделок подряд                    |
//| и время последней сделки закрытия убыточной позиции              |
//+------------------------------------------------------------------+
uint GetNumLosingTradesInRow(datetime &time_last_deal)
  {
//--- Выбираем всю историю
   if(!HistorySelect(0,TimeCurrent()))
      return(0);

//--- в цикле по списку исторических сделок получаем тикет очередной сделки
   uint res=0;
   uint total=HistoryDealsTotal();
   for(int i=(int)total-1; i>=0; i--)
     {
      ulong deal_ticket=HistoryDealGetTicket(i);
      if(deal_ticket>0)
        {
         //--- если сделка не на выход из позиции - идём к следующей
         ENUM_DEAL_ENTRY entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(deal_ticket,DEAL
         if(entry!=DEAL_ENTRY_OUT && entry!=DEAL_ENTRY_OUT_BY && entry!=DEAL_ENTRY_INO
            continue;
         //--- если результат закрытия позиции имеет положительный профит - прерываем
         if(!IsClosePositionWithLoss(deal_ticket))
            break;
         //--- увеличиваем счётчик идущих подряд сделок с отрицательной прибылью
         res++;
         //--- записываем в переменную максимальное время сделки (ищем последнюю)
         datetime deal_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
         if(deal_time>time_last_deal)
            time_last_deal=deal_time;
        }
     }

//--- Возвращаем количество идущих подряд убытков
   return(res);
  }
//+------------------------------------------------------------------+
//| Возвращает флаг закрытия позиции с убытком                       |
//+------------------------------------------------------------------+
bool IsClosePositionWithLoss(const ulong deal_ticket)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- получаем из сделки значения свойств, влияющих на прибыль
   double profit=HistoryDealGetDouble(deal_ticket,DEAL_PROFIT);
   double comission=HistoryDealGetDouble(deal_ticket,DEAL_COMMISSION);
   double swap=HistoryDealGetDouble(deal_ticket,DEAL_SWAP);
   double fee=HistoryDealGetDouble(deal_ticket,DEAL_FEE);

//--- возвращаем флаг того, что суммарное значение полученных свойств отрицательно
   return(profit+comission+swap+fee<0);
  }
```

© 2000-2025, MetaQuotes Ltd.

PeriodSeconds

Возвращает количество секунд в периоде.

```
int  PeriodSeconds(
   ENUM_TIMEFRAMES  period=PERIOD_CURRENT      // период графика
   );
```

Параметры

period=PERIOD_CURRENT

[in] Значение периода графика из перечисления ENUM_TIMEFRAMES. Если параметр не указан, то возвращается количество секунд текущего периода графика, на котором запущена программа.

Возвращаемое значение

Количество секунд в указанном периоде.

Пример:

```
//--- input parameters
input ENUM_TIMEFRAMES InpPeriod1 = PERIOD_CURRENT;    // First Period
input ENUM_TIMEFRAMES InpPeriod2 = PERIOD_M1;         // Second Period

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим количество секунд в периодах графика InpPeriod1 и InpPeriod2
   int sec1=PeriodSeconds(InpPeriod1);
   int sec2=PeriodSeconds(InpPeriod2);
//--- выведем полученные значения в журнал
   PrintFormat("Seconds in period %s: %lu, in period %s: %lu",TimeframeDescription(Inp
//--- рассчитаем, сколько баров периода графика InpPeriod2 содержится в баре с периодо
   int res=sec1/sec2;
   if(res==0)
      res=1;
//--- выведем полученное значение в журнал
   PrintFormat("One bar %s contains %d bars %s",TimeframeDescription(InpPeriod1),res,T
   /*
   Результат:
   Seconds in period M5: 300, in period M1: 60
   One bar M5 contains 5 bars M1
   */
  }
//+------------------------------------------------------------------+
//| Возвращает наименование таймфрейма                               |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
string TimeframeDescription(const ENUM_TIMEFRAMES period)
  {
   return(StringSubstr(EnumToString(period==PERIOD_CURRENT ? Period() : period), 7));
  }
```

Смотри также

_Period, Периоды графиков, Дата и время, Видимость объектов

© 2000-2025, MetaQuotes Ltd.

PlaySound

Воспроизводит звуковой файл.

```
bool  PlaySound(
   string  filename      // имя файла
   );
```

Параметры

filename

[in] Путь к звуковому файлу. Если filename=NULL, воспроизведение звука прекращается.

Возвращаемое значение

true - если звуковой файл найден, иначе возвращает false.

Примечание

Файл должен быть расположен в каталоге каталог_терминала\Sounds или его подкаталоге. Проигрываются только звуковые файлы в формате WAV.

Вызов PlaySound() с параметром NULL останавливает воспроизведение звука.

При работе в тестере стратегий функция PlaySound() не выполняется.

Пример:

```
#include <Trade\Trade.mqh>
#define MAGIC (123)

//--- input parameters
input string          InpFileNameOK  = "ok.wav";               // файл с озвучиванием
input string          InpFileNameErr = "timeout.wav";          // файл с озвучиванием
input ENUM_ORDER_TYPE InpOrderType   = ORDER_TYPE_BUY_LIMIT;   // тип ордера
input double          InpLots        = 0.1;                    // лоты

//--- global variables
CTrade ExtTrade;

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- устанавливаем магик и тип ордера по исполнению согласно настройкам символа
   ExtTrade.SetExpertMagicNumber(MAGIC);
   ExtTrade.SetTypeFillingBySymbol(Symbol());
//--- вызываем функцию установки ордера или открытия позиции с воспроизведением звука
   OrderSendWithAudio();
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Функция устанавливает ордер или открывает позицию                |
//| и воспроизводит звук успеха или ошибки                           |
//+------------------------------------------------------------------+
void OrderSendWithAudio(void)
  {
   bool    res=true;
   MqlTick tick= {};

   ResetLastError();
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      PlaySound(InpFileNameErr);
      return;
     }
//--- отправим запрос на сервер
   switch(InpOrderType)
     {
      case ORDER_TYPE_BUY :
         res=ExtTrade.Buy(InpLots);
         break;
      case ORDER_TYPE_BUY_LIMIT :
         res=ExtTrade.BuyLimit(InpLots,NormalizeDouble(tick.ask-100*Point(),Digits()))
         break;
      case ORDER_TYPE_BUY_STOP :
         res=ExtTrade.BuyStop(InpLots,NormalizeDouble(tick.ask+100*Point(),Digits()));
         break;
      case ORDER_TYPE_SELL :
         res=ExtTrade.Sell(InpLots);
         break;
      case ORDER_TYPE_SELL_LIMIT :
         res=ExtTrade.SellLimit(InpLots,NormalizeDouble(tick.bid+100*Point(),Digits())
         break;
      case ORDER_TYPE_SELL_STOP :
         res=ExtTrade.SellStop(InpLots,NormalizeDouble(tick.bid-100*Point(),Digits()))
         break;
      default :
         res=false;
     }
   if(!res)
      Print("Error ",GetLastError());
   Print(ExtTrade.ResultRetcodeDescription());

//--- если запрос принят, проигрываем звук ok.wav
   if(ExtTrade.ResultRetcode()==TRADE_RETCODE_DONE)
      PlaySound(InpFileNameOK);
   else
      PlaySound(InpFileNameErr);
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

Смотри также

Ресурсы

© 2000-2025, MetaQuotes Ltd.

Print

Печатает некоторое сообщение в журнал экспертов. Параметры могут иметь любой тип.

```
void  Print(
   argument,      // первое значение
   ...            // последующие значения
   );
```

Параметры

...

[in] Любые значения, разделенные запятыми. Количество параметров не может превышать 64.

Примечание

Массивы нельзя передавать в функцию Print(). Массивы должны печататься поэлементно.

Данные типа double выводятся с точностью до 16 десятичных цифр после точки, при этом данные могут выводиться либо в традиционном либо в научном формате - в зависимости от того, как запись будет наиболее компактна. Данные типа float выводятся с 5 десятичными цифрами после точки. Для вывода вещественных чисел с другой точностью либо в явно указанном формате необходимо использовать функцию PrintFormat().

Данные типа bool выводятся в виде строк "true" или "false". Даты выводятся в виде YYYY.MM.DD HH:MI:SS. Для вывода даты в другом формате необходимо использовать функцию TimeToString(). Данные типа color выводятся либо в виде строки R,G,B, либо в виде названия цвета, если этот цвет присутствует в наборе цветов.

При работе в тестере стратегий в режиме оптимизации функция Print() не выполняется.

Пример:

```
void OnStart()
  {
//--- выведем DBL_MAX с помощью Print(), это равносильно PrintFormat(%%.16G,DBL_MAX)
   Print("---- как выглядит DBL_MAX -----");
   Print("Print(DBL_MAX)=",DBL_MAX);
//--- теперь выведем число DBL_MAX с помощью PrintFormat()
   PrintFormat("PrintFormat(%%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- Вывод в журнал "Эксперты"
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- посмотрим как выводится тип float
   float c=(float)M_PI; // нужно явно приводить к целевому типу
   Print("c=",c, "    Pi=",M_PI, "    (float)M_PI=",(float)M_PI);
// c=3.14159    Pi=3.141592653589793    (float)M_PI=3.14159

//--- покажем, что может произойти при арифметических операциях над вещественными типа
   double a=7,b=200;
   Print("---- перед арифметическими операциями");
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("a=",a,"   b=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- разделим a на b (7/200)
   a=a/b;
//--- теперь как будто восстановим значение в переменной b
   b=7.0/a; // ожидается, что b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 - но это не так
//--- выведем вновь вычисленное значение b
   Print("----- после арифметических операций");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- вывод в журнал "Эксперты"
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.9999999999999716 (видим, что на самом деле b уже не

//--- создадим очень маленькое значение epsilon=1E-013
   double epsilon=1e-13;
   Print("---- создадим очень маленькое число");
   Print("epsilon=",epsilon); // получим   epsilon=1E-013
//--- теперь вычтем эпсилон из числа b и выведем снова значение в журнал "Эксперты"
   b=b-epsilon;
//--- выводим двумя способами
   Print("---- после вычитания epsilon из переменной b");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- вывод в журнал "Эксперты"
// Print(b)=199.9999999999999  (теперь значение b после вычитания эпсилон не может окр
// Print(DoubleToString(b,16))=199.9999999999998578
//     (теперь значение b после вычитания эпсилон не может округлиться до 200)
  }
```

Смотри также

DoubleToString, StringFormat

© 2000-2025, MetaQuotes Ltd.

PrintFormat

Форматирует и печатает наборы символов и значений в журнал экспертов в соответствии с заданным форматом.

```
void  PrintFormat(
   string format_string,    // форматная строка
   ...                      // значения простых типов
   );
```

Параметры

format_string

[in] Строка формата состоит из обычных символов и, если за строкой формата следуют аргументы, еще и спецификации формата.

...

[in] Любые значения простых типов, разделенные запятыми. Общее количество параметров не может превышать 64, включая форматную строку.

Возвращаемое значение

Строка.

Примечание

При работе в тестере стратегий в режиме оптимизации функция PrintFormat() не выполняется.

Количество, порядок и тип параметров должны точно соответствовать составу спецификаторов, в противном случае результат печати неопределён. Вместо функции PrintFormat() можно использовать функцию printf().

Если за строкой формата следуют еще параметры, то эта строка должна содержать спецификации формата, определяющие формат вывода этих параметров. Спецификация формата всегда начинается с символа знака процента (%).

Строка формата читается слева направо. Когда встречается первая спецификация формата (если она есть), то значение первого параметра после строки формата преобразовывается и выводится согласно заданной спецификации. Вторая спецификация формата вызывает преобразование и вывод второго параметра и так далее, до конца строки формата.

Спецификация формата имеет следующую форму:

%[flags][width][.precision][{h | l | ll | I32 | I64}]type

Каждое поле форматной спецификации является либо простым символом, либо числом, обозначающим обычную форматную опцию. Простейшая спецификация формата содержит только знак процента (%) и символ, определяющий тип выводимого параметра (например %s). Если требуется в форматной строке вывести символ знак процента, то необходимо использовать форматную спецификацию %%.

flags

© 2000-2025, MetaQuotes Ltd.

Флаг	Описание	Поведение по умолчанию
- (минус)	Выравнивание по левому краю в пределах заданной ширины	Выравнивание по правому краю
+ (плюс)	Вывод знака + или - для значений знаковых типов	Знак выводится только если значение отрицательное
0 (ноль)	Перед выводимом значением добавляются нули в пределах заданной ширины. Если указан флаг **0** с целочисленным форматом (**i**, **u**, **x**, **X**, **o**, **d**) и задана спецификация точности (например, %04.d), то 0 игнорируется.	Ничего не вставляется
пробел	Перед выводимым значением ставится пробел, если значение является знаковым и положительным	Пробелы не вставляются
#	Если используется совместно с форматом **o**, **x** или **X**, то перед выводимым значением добавляется 0, 0x или 0X соответственно.	Ничего не вставляется
	Если используется совместно с форматом **e**, **E**, **a** или **A**, то значение всегда выводится с десятичной точкой.	Десятичная точка выводится только если есть ненулевая дробная часть.
	Если используется совместно с форматом **g** или **G**, флаг определяет наличие десятичной точки в выводимом значении и препятствует отсечению ведущих нулей. Флаг # игнорируется при совместном использовании с форматами **c**, **d**, **i**, **u**, **s**.	Десятичная точка выводится только если есть ненулевая дробная часть. Ведущие нули отсекаются

width

Неотрицательное десятичное число, которое задает минимальное число выводимых символов отформатированного значения. Если количество выводимых символов меньше указанной ширины, то добавляется соответствующее количество пробелов слева или справа в зависимости от выравнивания (флаг -). При наличии флага ноль (0), перед выводимым значением добавляется соответствующее количество нулей. Если число выводимых символов больше заданной ширины, то выводимое значение никогда не усекается.

Если в качестве ширины указана звездочка (*), то в списке передаваемых параметров на соответствующем месте должно быть значение типа int, которое будет использовано для указания ширины выводимого значения.

precision

© 2000-2025, MetaQuotes Ltd.

Неотрицательное десятичное число, которое определяет точность вывода - количество цифр после десятичной точки. В отличие от спецификации ширины, спецификация точности может отсекать часть дробного значения с округлением или без округления.

Для разных форматных типов (**type**) спецификация точности применяется по-разному.

Типы	Описание	Поведение по умолчанию
a, A	Спецификация точности указывает количество знаков после десятичной точки	Точность по умолчанию - 6.
c, C	Не применяется	
d, i, u, o, x, X	Указывает минимальное число выводимых цифр. Если количество цифр в соответствующем параметре меньше указанной точности, то выводимое значение дополняется слева нулями. Выводимое значение не обрезается, если количество выводимых цифр больше указанной точности	Точность по умолчанию - 1.
e, E, f	Указывает число выводимых цифр после десятичной точки. Последняя выводимая цифра округляется	Точность по умолчанию - 6. Если указана точность 0 или дробная часть отсутствует, то десятичная точка не выводится.
g, G	Указывает максимальное число значимых цифр	Выводится 6 значимых цифр.
s	Указывает количество выводимых символов строки. Если длина строки превышает значение точности, то строка усекается на выводе	Выводится вся строка

```
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

Спецификации размеров данных, передаваемых в качестве параметра.

© 2000-2025, MetaQuotes Ltd.

Тип параметра	Используемый префикс	Совместный спецификатор типа
int	l (маленькая L)	d, i, o, x, or X
uint	l (маленькая L)	o, u, x, or X
long	ll (две маленькие 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

type

Спецификатор типа является единственным обязательным полем для форматированного вывода.

Символ	Тип	Выводной формат
c	int	Символ типа short (Unicode)
C	int	Символ типа char (ANSI)
d	int	Знаковое десятичное целое
i	int	Знаковое десятичное целое
o	int	Беззнаковое восьмеричное целое
u	int	Беззнаковое десятичное целое
x	int	Беззнаковое шестнадцатиричное целое с использованием "abcdef"
X	int	Беззнаковое шестнадцатиричное целое с использованием "ABCDEF"
e	double	Вещественное значение в формате [-]d.dddd **e** [sign]ddd, где d - одна десятичная цифра, dddd - одна или больше десятичных цифр, ddd - трехзначное число, определяющее размер экспоненты, sign - знак плюс или минус
E	double	Идентично формату e, за исключением того, что знак экспоненты вывоится большой буквой (**E** вместо e)
f	double	Вещественное значение в формате [-]dddd.dddd, где dddd - одна или больше десятичных цифр. Количество выводимых знаков перед

© 2000-2025, MetaQuotes Ltd.

Символ	Тип	Выводной формат
		десятичной точкой зависит от величины значения числа. Количество знаков после десятичной точки зависит от требуемой точности.
g	double	Вещественное значение, выводимое в формате **f** или **e**, в зависимости от того какой вывод будет компактнее.
G	double	Вещественное значение, выводимое в формате **f** или **E**, в зависимости от того какой вывод будет компактнее.
a	double	Вещественное значение в формате [-]0xh.hhhh **p**±dd, где h.hhhh - мантисса в виде шестнадцатиричных цифр с использованием "abcdef", dd - одна или более цифр экспоненты. Количество знаков после запятой определяется спецификацией точности
A	double	Вещественное значение в формате [-]0xh.hhhh **P**±dd, где h.hhhh - мантисса в виде шестнадцатиричных цифр с использованием "ABCDEF", dd - одна или более цифр экспоненты. Количество знаков после запятой определяется спецификацией точности
s	string	Вывод строки

Вместо функции PrintFormat() можно использовать функцию printf().

Пример:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- имя торгового сервера
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- номер торговго счета
   int login=(int)AccountInfoInteger(ACCOUNT_LOGIN);
//--- вывод значения long
   long leverage=AccountInfoInteger(ACCOUNT_LEVERAGE);
   PrintFormat("%s %d: плечо = 1:%I64d",
               server,login,leverage);
//--- валюта депозита
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- вывод значения double с 2 цифрами после десятичной точки
   double equity=AccountInfoDouble(ACCOUNT_EQUITY);
   PrintFormat("%s %d: размер собственных средств на счете = %.2f %s",
               server,login,equity,currency);
//--- вывод значения double с обязательным выводом знака +/-
   double profit=AccountInfoDouble(ACCOUNT_PROFIT);
   PrintFormat("%s %d: текущий результат по открытым позициям = %+.2f %s",
               server,login,profit,currency);
//--- вывод значения double с переменным количеством цифр после десятичной точки
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   string format_string=StringFormat("%%s: значение одного пункта  = %%.%df",_Digits);
   PrintFormat(format_string,_Symbol,point_value);
//--- вывод значения int
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   PrintFormat("%s: текущий спред в пунктах = %d ",
               _Symbol,spread);
//--- вывод значения double в научном формате с плавающей запятой и точностью 17 знача
   PrintFormat("DBL_MAX = %.17e",DBL_MAX);
//--- вывод значения double в научном формате с плавающей запятой и точностью 17 знача
   PrintFormat("EMPTY_VALUE = %.17e",EMPTY_VALUE);
//--- вывод через PrintFormat() с точностью по умолчанию
   PrintFormat("PrintFormat(EMPTY_VALUE) = %e",EMPTY_VALUE);
//--- простой вывод через Print()
   Print("Print(EMPTY_VALUE) = ",EMPTY_VALUE);
/* результат выполнения
   MetaQuotes-Demo 1889998: плечо = 1:100
   MetaQuotes-Demo 1889998: размер собственных средств на счете = 22139.86 USD
   MetaQuotes-Demo 1889998: текущий результат по открытым позициям = +174.00 USD
   EURUSD: значение одного пункта  = 0.00001
   EURUSD: текущий спред в пунктах = 12
   DBL_MAX = 1.79769313486231570e+308
   EMPTY_VALUE = 1.79769313486231570e+308
   PrintFormat(EMPTY_VALUE) = 1.797693e+308
   Print(EMPTY_VALUE) = 1.797693134862316e+308
*/
  }
```

Смотри также

StringFormat, DoubleToString, Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

ResetLastError

Устанавливает значение предопределенной переменной _LastError в ноль.

```
void  ResetLastError();
```

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Необходимо отметить, что функция GetLastError() не обнуляет переменную _LastError. Обычно вызов функции ResetLastError() производится перед вызовом функции, после которой проверяется возникновение ошибки.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- сбрасываем код последней ошибки перед вызовом функции,
//--- так как в противном случае GetLastError() может вернуть код предыдущей ошибки
   long lres=SymbolInfoInteger("123456",SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
   lres=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
   ResetLastError();
   lres=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

ResourceCreate

Создает ресурс изображения на основе набора данных. Существует два варианта функции:
Создание ресурса на основе файла

```
bool  ResourceCreate(
   const string      resource_name,        // имя ресурса
   const string      path                  // относительный путь к файлу
   );
```

Динамическое создание ресурса на основе массива пикселей

```
bool  ResourceCreate(
   const string      resource_name,        // имя ресурса
   const uint&       data[],               // набор данных в виде массива
   uint              img_width,            // ширина создаваемой картинки-ресурса
   uint              img_height,           // высота создаваемой картинки-ресурса
   uint              data_xoffset,         // смещение левого верхнего угла создаваемой
   uint              data_yoffset,         // смещение левого верхнего угла создаваемой
   uint              data_width,           // общая ширина изображения на основе набора
   ENUM_COLOR_FORMAT color_format          // способ обработки цвета
   );
```

Параметры

resource_name

[in] Имя ресурса.

path

[in] Относительный путь к файлу, содержащему данные для ресурса. Если путь начинается с обратной косой черты "\" (пишется "\\"), то файл ищется относительно папки **каталог_данных_терминала**\MQL5\. Если обратной косой черты нет, то ресурс ищется относительно расположения EX5-файла, из которого вызывается функция.

data[][]

[in] Одномерный или двумерный массив для создания полного изображения.

img_width

[in] Ширина прямоугольной области изображения в пикселях для помещения в ресурс в виде картинки. Не может быть больше значения *data_width*.

img_height

[in] Высота прямоугольной области изображения в пикселях для помещения в ресурс в виде картинки.

data_xoffset

[in] Смещение в пикселях прямоугольной области изображения по горизонтали вправо.

data_yoffset

[in] Смещение в пикселях прямоугольной области изображения по вертикали вниз.

data_width

© 2000-2025, MetaQuotes Ltd.

[in] Требуется только для одномерных массивов и означает полную ширину создаваемого изображения из набора данных. Если *data_width*=0, то подразумевается равным *img_width*. Для двумерных массивов данный параметр игнорируется и принимается равным второй размерности массива *data[]*.

color_format

[in] Способ обработки цвета из перечисления ENUM_COLOR_FORMAT.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4015 - ERR_RESOURCE_NAME_DUPLICATED (совпадение имен динамического и статического ресурсов),
- 4016 - ERR_RESOURCE_NOT_FOUND (ресурс не найден),
- 4017 - ERR_RESOURCE_UNSUPPORTED_TYPE (тип ресурса не поддерживается),
- 4018 - ERR_RESOURCE_NAME_IS_TOO_LONG (слишком длинное имя ресурса).

Примечание

Если второй вариант функции вызывается для создания одного и того же ресурса с разными параметрами ширины, высоты и сдвига, то новый ресурс не пересоздается, а просто обновляется существующий.

Первый вариант функции позволяет загружать из файлов картинки и звуки, второй вариант предназначен только для динамического создания изображений.

Картинки должны быть в формате BMP с глубиной цвета 24 или 32 бита, звуки могут быть только в формате WAV. Размер ресурса не должен превышать 16 Mb.

ENUM_COLOR_FORMAT

Идентификатор	Описание
COLOR_FORMAT_XRGB_NOALPHA	Компонента альфа-канала игнорируется
COLOR_FORMAT_ARGB_RAW	Компоненты цвета не обрабатываются терминалом (должны быть корректно заданы пользователем)
COLOR_FORMAT_ARGB_NORMALIZE	Компоненты цвета обрабатываются терминалом

Пример:

```
//+------------------------------------------------------------------+
//| Обновляет данные графического ресурса                            |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- если переданы нулевые размеры - уходим
   if(width==0 || height==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
//--- обновляем ресурсные данные и перерисовываем график
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZ
      ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявляем параметры графического ресурса
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- устанавливаем размер массива пикселей
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- заполняем массив пикселей прозрачным цветом и на его основе создаём графический
   ArrayInitialize(rc_data,0x00FFFFFF);
   if(!ResourceCreate(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- проверим созданный графический ресурс.
//--- получаем данные времени и цены текущего бара
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- создаём объект "Рисунок" на координатах цены и времени последнего тика
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- устанавливаем созданному объекту-рисунку ширину и высоту, равные ширине и высоте
```

© 2000-2025, MetaQuotes Ltd.

```
//--- устанавливаем точку привязки объекта по его центру.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- указываем для объекта-рисунка в качестве файла изображения ранее созданный графи
//--- в этом случае для указания имени используемого графического ресурса нужно перед
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- установим цвет DodgerBlue с прозрачностью 200
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- заполним установленным цветом весь массив пикселей графического ресурса
   ArrayInitialize(rc_data,clr);
//--- обновим данные графического ресурса
   Update(rc_name,rc_data,rc_width,rc_height,true);
  }
```

Смотри также

Ресурсы, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceFree

Удаляет динамически созданный ресурс (освобождает занятую ресурсом память).

```
bool  ResourceFree(
   const string  resource_name      // имя ресурса
   );
```

Параметры

resource_name

[in] Имя ресурса, должно начинаться с "::".

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция ResourceFree() позволяет разработчику mql5-программы управлять потреблением памяти при активной работе с ресурсами. Графические объекты, привязанные к удаляемому из памяти ресурсу, будут отображаться правильно и после его удаления. Но вновь созданные графические объекты (OBJ_BITMAP и OBJ_BITMAP_LABEL) уже не смогут использовать удалённый ресурс.

Функция удаляет только динамические ресурсы, созданные данной программой.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявляем параметры графического ресурса
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- устанавливаем размер массива пикселей
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- заполняем массив пикселей прозрачным цветом и на его основе создаём графический
   ArrayInitialize(rc_data,0x00FFFFFF);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ResourceCreate(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- проверим созданный графический ресурс.
//--- получаем данные времени и цены текущего бара
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- создаём объект "Рисунок" на координатах цены и времени последнего тика
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- устанавливаем созданному объекту-рисунку ширину и высоту, равные ширине и высоте
//--- устанавливаем точку привязки объекта по его центру.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- указываем для объекта-рисунка в качестве файла изображения ранее созданный графи
//--- в этом случае для указания имени используемого графического ресурса нужно перед
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- Установим цвет DodgerBlue с прозрачностью 200
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- заполним установленным цветом весь массив пикселей графического ресурса
   ArrayInitialize(rc_data,clr);
//--- обновим данные графического ресурса
   Update(rc_name,rc_data,rc_width,rc_height,true);

//--- подождём три секунды и сменим цвет изображения
   Print("Wait 3 seconds before changing color");
   Sleep(3000);
//--- установим цвет OrangeRed с прозрачностью 200
   Print("Change color");
   clr=ColorToARGB(clrOrangeRed,200);
//--- заполним установленным цветом весь массив пикселей графического ресурса
   ArrayInitialize(rc_data,clr);
//--- обновим данные графического ресурса
   Update(rc_name,rc_data,rc_width,rc_height,true);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ждём три секунды и освобождаем графический ресурс
   Print("Wait 3 seconds before ResourceFree()");
   Sleep(3000);
   bool res=ResourceFree("::"+rc_name);
   Print("ResourceFree: ",res);

//--- пробуем поменять цвет после освобождения ресурса
   Print("Trying to change color to GreenYellow after ResourceFree()");
//--- установим цвет GreenYellow с прозрачностью 200
   clr=ColorToARGB(clrGreenYellow,200);
//--- заполним установленным цветом весь массив пикселей графического ресурса
   ArrayInitialize(rc_data,clr);
//--- обновим данные графического ресурса (рисунок остаётся, но цвет изменить уже нево
   Update(rc_name,rc_data,rc_width,rc_height,true);
   Print("The color has not changed because the resource has been released");

//--- ждём три секунды и удаляем объект-рисунок
   Print("Wait 3 seconds before deleting the Bitmap object");
   Sleep(3000);
   Print("Delete Bitmap object");
   ObjectDelete(0,obj_name);
  }
//+------------------------------------------------------------------+
//| Обновляем данные графического ресурса                            |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- если переданы нулевые размеры - уходим
   if(width==0 || height==0)
      return;
//--- обновляем ресурсные данные и перерисовываем график
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZ
      ChartRedraw();
  }
```

Смотри также

Ресурсы, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceReadImage

Читает данные графического ресурса, созданного функцией ResourceCreate() или сохраненного в EX5-файле при компиляции.

```
bool  ResourceReadImage(
   const string      resource_name,       // имя графического ресурса для чтения
   uint&             data[],              // массив для получения данных из ресурса
   uint&             width,               // для получения ширины картинки в ресурсе
   uint&             height,              // для получения высоты картинки в ресурсе
   );
```

Параметры

resource_name

[in] Имя графического ресурса, содержащего изображение. Для доступа к собственным ресурсам указывается в коротком виде "::resourcename". Если же необходимо загрузить ресурс из скомпилированного EX5-файла, то необходимо имя в полном виде с указанием пути относительно папки MQL5, имени файла и имени ресурса - "path\\filename.ex5::resourcename".

data[][]

[in] Одномерный или двумерный массив для получения данных из графического ресурса.

img_width

[out] Ширина картинки графического ресурса в пикселях.

img_height

[out] Высота картинки графического ресурса в пикселях.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Если на основании массива *data[]* в дальнейшем необходимо создать графический ресурс, то следует использовать формат цвета COLOR_FORMAT_ARGB_NORMALIZE или COLOR_FORMAT_XRGB_NOALPHA.

Если массив *data[]* является двумерным и его вторая размерность меньше размера X(width) графического ресурса, то функция ResourceReadImage() вернет false и чтение не будет произведено. Но при этом, если ресурс существует, то в параметры width и height возвращаются актуальные размеры картинки. Это позволит сделать еще одну попытку получения данных из ресурса.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- объявляем параметры графического ресурса
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- устанавливаем размер массива пикселей
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- заполняем массив пикселей прозрачным цветом и на его основе создаём графический
   ArrayInitialize(rc_data,0x00FFFFFF);
   if(!ResourceCreate(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- проверим созданный графический ресурс.
//--- получаем данные времени и цены текущего бара
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- создаём объект "Рисунок" на координатах цены и времени последнего тика
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- устанавливаем созданному объекту-рисунку ширину и высоту, равные ширине и высоте
//--- устанавливаем точку привязки объекта по его центру.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- указываем для объекта-рисунка в качестве файла изображения ранее созданный графи
//--- В этом случае для указания имени используемого графического ресурса нужно перед
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- установим цвет DodgerBlue с прозрачностью 200
```

© 2000-2025, MetaQuotes Ltd.

```
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- заполним установленным цветом весь массив пикселей графического ресурса
   ArrayInitialize(rc_data,clr);
//--- обновим данные графического ресурса
   Update(rc_name,rc_data,rc_width,rc_height,true);

//--- ждём 3 секунды перед чтением данных графического ресурса
   Print("Wait 3 seconds before ResourceReadImage()");
   Sleep(3000);
//--- читаем изображение из ресурса в новый массив пикселей
   uint rc_data_copy[];
   uint w=0,h=0;
   ResetLastError();
   if(!ResourceReadImage("::"+rc_name,rc_data_copy,w,h))
     {
      Print("ResourceReadImage() failed. Error code: ",GetLastError());
      return;
     }

//--- установим цвет OrangeRed с прозрачностью 200
   clr=ColorToARGB(clrOrangeRed,200);
//--- заполним установленным цветом весь массив пикселей и на его основе создадим новы
   ArrayInitialize(rc_data_copy,clr);
   if(!ResourceCreate(rc_name+"Copy",rc_data_copy,rc_width,rc_height,0,0,0,COLOR_FORMA
     {
      Print("New ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created new recource array: ",rc_data_copy.Size());

//--- создаём объект "Графическая метка" на координатах цены и времени последнего тика
   string obj_name2="BitmapLabel";
   if(!ObjectCreate(0,obj_name2,OBJ_BITMAP_LABEL,0,0,0))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- получаем экранные координаты по ранее полученным цене и времени
   int x=0,y=0;
   if(!ChartTimePriceToXY(0,0,tick.time,tick.bid,x,y))
     {
      Print("New ChartTimePriceToXY() failed. Error code: ",GetLastError());
      return;
     }
//--- устанавливаем созданному объекту-графической метке ширину и высоту, равные ширин
//--- устанавливаем точку привязки объекта по его центру.
   ObjectSetInteger(0,obj_name2,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name2,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name2,OBJPROP_ANCHOR,ANCHOR_LEFT_UPPER);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(0,obj_name2,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(0,obj_name2,OBJPROP_YDISTANCE,y);
//--- указываем для объекта-графической метки в качестве файла изображения скопирован
//--- в этом случае для указания имени используемого графического ресурса нужно перед
   ObjectSetString(0,obj_name2,OBJPROP_BMPFILE,"::"+rc_name+"Copy");

//--- меняем цвет нового объекта-графической метки
   Print("Wait 3 seconds before changing color to GreenYellow");
   Sleep(3000);
//--- установим цвет GreenYellow с прозрачностью 200
   clr=ColorToARGB(clrGreenYellow,200);
//--- заполним установленным цветом весь массив пикселей нового графического ресурса
   ArrayInitialize(rc_data_copy,clr);
//--- обновим данные графического ресурса
   Update(rc_name+"Copy",rc_data_copy,rc_width,rc_height,true);

//--- ждём три секунды и удаляем ресурсы и оба объекта
   Print("Wait 3 seconds before deleting both objects");
   Sleep(3000);
   Print("Deleting Resource and all Bitmap objects");
   ResourceFree("::"+rc_name);
   ResourceFree("::"+rc_name+"Copy");
   ObjectDelete(0,obj_name);
   ObjectDelete(0,obj_name2);
  }
//+------------------------------------------------------------------+
//| Обновляем данные графического ресурса                            |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- если переданы нулевые размеры - уходим
   if(width==0 || height==0)
      return;
//--- обновляем ресурсные данные и перерисовываем график
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALI
      ChartRedraw();
  }
```

Смотри также

Ресурсы, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceSave

Сохраняет ресурс в указанный файл.

```
bool  ResourceSave(
   const string  resource_name      // имя ресурса
   const string  file_name          // имя файла
   );
```

Параметры

resource_name

[in] Имя ресурса, должно начинаться с "::".

file_name

[in] Имя файла относительно MQL5\Files.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция всегда перезаписывает файл и создаёт при необходимости все промежуточные подкаталоги в имени файла в случае их отсутствия.

Пример:

```
//--- параметры графического ресурса
string ExtResName="Resource";
int    ExtResWidth=100;
int    ExtResHeight=100;
uint   ExtResData[];
int    ExtResSize=ExtResWidth*ExtResHeight;

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   ResetLastError();
//--- устанавливаем размер массива пикселей
   if(ArrayResize(ExtResData,ExtResSize)!=ExtResSize)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- заполняем массив пикселей прозрачным цветом и на его основе создаём графический
   ArrayInitialize(ExtResData,0x00FFFFFF);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ResourceCreate(ExtResName,ExtResData,ExtResWidth,ExtResHeight,0,0,0,COLOR_FORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",ExtResData.Size());

//--- проверим созданный графический ресурс.
//--- получаем данные времени и цены текущего бара
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- создаём объект "Рисунок" на координатах цены и времени последнего тика
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- устанавливаем созданному объекту-рисунку ширину и высоту, равные ширине и высоте
//--- устанавливаем точку привязки объекта по его центру.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,ExtResWidth);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,ExtResHeight);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- указываем для объекта-рисунка в качестве файла изображения ранее созданный графи
//--- в этом случае для указания имени используемого графического ресурса нужно перед
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+ExtResName);

//--- установим цвет GreenYellow с прозрачностью 200
   uint clr=ColorToARGB(clrGreenYellow,200);
//--- заполним установленным цветом весь массив пикселей графического ресурса
   ArrayInitialize(ExtResData,clr);
//--- нарисуем сетку цветом DodgerBlue
   for(int x=0;x<ExtResWidth;x+=9)
      LineVertical(x,0,ExtResHeight,ColorToARGB(clrDodgerBlue,200));
   for(int y=0;y<ExtResHeight;y+=9)
      LineHorizontal(0,ExtResWidth,y,ColorToARGB(clrDodgerBlue,200));
//--- обновим данные графического ресурса
   Update(ExtResName,ExtResData,ExtResWidth,ExtResHeight,true);

//--- подождём 3 секунды перед удалением ресурса и графического объекта
   Print("Wait 3 seconds before deleting the resource and graphic object");
   Sleep(3000);

//--- сохраним ресурс в файле
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ResourceSave("::"+ExtResName,"ResourceSave\\"+ExtResName+".bmp"))
     {
      Print("ResourceSave() failed. Error code: ",GetLastError());
      return;
     }
//--- удалим ресурс и графический объект-рисунок
   if(!ResourceFree("::"+ExtResName))
     {
      Print("ResourceFree() failed. Error code: ",GetLastError());
      return;
     }
   if(!ObjectDelete(0,obj_name))
     {
      Print("ObjectDelete() failed. Error code: ",GetLastError());
      return;
     }

//--- теперь создадим новый объект "Графическая метка" на координатах цены и времени п
//--- получим экранные координаты по ранее полученным цене и времени последнего тика
   int x=0,y=0;
   if(!ChartTimePriceToXY(0,0,tick.time,tick.bid,x,y))
     {
      Print("ChartTimePriceToXY() failed. Error code: ",GetLastError());
      return;
     }
   obj_name="BitmapLabel";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP_LABEL,0,0,0))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- устанавливаем точку привязки объекта по его центру.
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- устанавливаем объекту-графической метке координаты цены и времени последнего ти
   ObjectSetInteger(0,obj_name,OBJPROP_XDISTANCE,x-ExtResWidth);
   ObjectSetInteger(0,obj_name,OBJPROP_YDISTANCE,y);
//--- указываем для объекта-графической метки в качестве файла изображения ранее сохра
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"\\Files\\ResourceSave\\"+ExtResName+".b
//--- обновляем график
   ChartRedraw();

//--- подождём 3 секунды перед удалением объекта-графической метки
   Print("Wait 3 seconds before deleting the new graphic object");
   Sleep(3000);
   if(!ObjectDelete(0,obj_name))
      Print("ObjectDelete() failed. Error code: ",GetLastError());
  }
//+------------------------------------------------------------------+
//| Обновляет данные графического ресурса                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- если переданы нулевые размеры - уходим
   if(width==0 || height==0)
      return;
//--- обновляем ресурсные данные и перерисовываем график
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALIZ
      ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Рисует вертикальную линию                                        |
//+------------------------------------------------------------------+
void LineVertical(int x,int y1,int y2,const uint clr)
  {
   int tmp;
//--- сортировка по Y
   if(y1>y2)
     {
      tmp=y1;
      y1 =y2;
      y2 =tmp;
     }
//--- линия за границами картинки
   if(y2<0 || y1>=ExtResHeight || x<0 || x>=ExtResWidth)
      return;
//--- остаемся внутри границ картинки
   if(y1<0)
      y1=0;
   if(y2>=ExtResHeight)
      y2=ExtResHeight-1;
//--- рисуем линию
   int index=y1*ExtResWidth+x;
   for(int i=y1; i<=y2; i++,index+=ExtResWidth)
      ExtResData[index]=clr;
  }
//+------------------------------------------------------------------+
//| Рисует горизонтальную линию                                      |
//+------------------------------------------------------------------+
void LineHorizontal(int x1,int x2,int y,const uint clr)
  {
   int tmp;
//--- сортировка по X
   if(x1>x2)
     {
      tmp=x1;
      x1 =x2;
      x2 =tmp;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- линия за границами картинки
   if(x2<0 || x1>=ExtResWidth || y<0 || y>=ExtResHeight)
      return;
//--- остаемся внутри границ картинки
   if(x1<0)
      x1=0;
   if(x2>=ExtResWidth)
      x2=ExtResWidth-1;
//--- рисуем линию
   ArrayFill(ExtResData,y*ExtResWidth+x1,(x2-x1)+1,clr);
  }
```

Смотри также

Ресурсы, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

SetReturnError

Устанавливает код возврата, который вернёт процесс терминала при завершение работы.

```
void  SetReturnError(
   int ret_code      // код завершения клиентского терминала
   );
```

Параметры

ret_code

[in] Код возврата, который будет возвращен процессом клиентского терминала при завершении работы.

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Установка заданного кода возварта *ret_code* с помощью функции SetReturnError() будет полезна для анализа причин программного прекращения работы при запуске терминала из командной строки.

Функция SetReturnError() в отличие от TerminalClose() не завершает работу терминала, она только устанавливает код возврата, который вернёт процесс терминала при его завершении.

Если функция SetReturnError() вызывалась многократно и/или из разных MQL5 программ, то терминал вернёт последний установленный код возврата.

Выставленный код будет возвращаться при завершении процесса терминала кроме следующих случаев:

- возникла критическая ошибка во время выполнения;
- была вызвана функции TerminalClose(int ret_code), которая отдает терминалу команду на завершение работы с заданным кодом.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   matrix matrix_a =
     {
        {-3.474589, 1.106384, -9.091977,-3.925227 },
        {-5.522139, 2.366887,-15.162351,-6.357512 },
        { 8.394926,-2.960067, 22.292115, 9.524129 },
        { 7.803242,-2.080287, 19.217706, 8.186645 }
     };
   matrix matrix_l(4,4);
   matrix matrix_u(4,4);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- LU декомпозиция
   matrix_a.LU(matrix_l,matrix_u);

//--- проверим правильность A = L * U
   matrix matrix_lu=matrix_l.MatMul(matrix_u);
   int    compare_errors=(int)matrix_a.Compare(matrix_lu,1e-29);
   Print("MatrixCompare errors=",compare_errors);

//--- клиентский терминал при своём завершении вернёт количество ошибок сравнения дву
   SetReturnError(compare_errors);
  }
```

Смотри также

Выполнение программ, Ошибки выполнения, Причины деинициализации, TerminalClose

© 2000-2025, MetaQuotes Ltd.

SetUserError

Устанавливает предопределенную переменную _LastError в значение, равное ERR_USER_ERROR_FIRST + user_error

```
void  SetUserError(
   ushort user_error,    // номер ошибки
   );
```

Параметры

user_error

[in] Номер ошибки, устанавливаемый пользователем.

Возвращаемое значение

Нет возвращаемого значения.

Примечание

После того, как была выставлена ошибка при помощи функции SetUserError(user_error), функция GetLastError() вернет значение, равное ERR_USER_ERROR_FIRST + user_error.

Пример:

```
void OnStart()
  {
//--- установим номер ошибки 65537=(ERR_USER_ERROR_FIRST +1)
   SetUserError(1);
//--- получим код последней ошибки
   Print("GetLastError = ",GetLastError());
/*
   Результат
   GetLastError = 65537
*/
  }
```

© 2000-2025, MetaQuotes Ltd.

Sleep

Функция задерживает выполнение текущего эксперта или скрипта на определенный интервал.

```
void  Sleep(
   int  milliseconds      // интервал
   );
```

Параметры

milliseconds

[in] Интервал задержки в миллисекундах.

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Функцию Sleep() нельзя вызывать из пользовательских индикаторов, так как индикаторы выполняются в интерфейсном потоке и не должны его тормозить. В функцию встроена проверка состояния флага остановки эксперта каждую 0.1 секунды.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- выведем в комментарии на графике обратный отсчёт от 10 до 1
   for(int i=10; i>0 && !IsStopped(); i--)
     {
      Comment(StringFormat("Wait %u seconds",i));
      Sleep(1000);
     }
//--- напишем в "прибывающем" комментарии текст, описывающий назначение скрипта
   string text="This was a test showing how the Sleep() function works";
   string mess="";
   for(int i=0; i<(int)text.Length(); i++)
     {
      mess+=ShortToString(text.GetChar(i));
      Sleep(100);
      Comment(mess);
     }
//--- попрощаемся...
   Sleep(1000);
   for(int i=0; i<6; i++)
     {
      mess=(i % 2 == 0 ? "" : "  Bye!");
```

© 2000-2025, MetaQuotes Ltd.

```
      Comment(mess);
      Sleep(300);
     }
//--- сотрём текст на графике
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalClose

Посылает терминалу команду на завершение работы.

```
bool  TerminalClose(
   int ret_code      // код завершения клиентского терминала
   );
```

Параметры

ret_code

[in] Код возврата, возвращаемый процессом клиентского терминала при завершении работы.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Функция TerminalClose() не производит немедленной остановки работы терминала, она просто посылает терминалу команду на завершение.

В коде советника, вызвавшего TerminalClose(), должны быть сделаны все приготовления для немедленного завершения работы (например, должны быть штатным образом закрыты все ранее открытые файлы). Сразу после вызова этой функции должен идти оператор return.

Параметр *ret_code* позволяет указывать нужный код возврата для анализа причин программного прекращения работы терминала при его запуске из командной строки.

Пример:

```
//--- input parameters
input int  tiks_before=500; // количество тиков до завершения
input int  pips_to_go=15;   // расстояние в пипсах
input int  seconds_st=50;   // сколько секунд даем эксперту
//--- globals
datetime   launch_time;
int        tick_counter=0;
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(__FUNCTION__," reason code = ",reason);
   Comment("");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static double first_bid=0.0;
```

© 2000-2025, 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;
     }
//--- ход цены в пунктах
   distance=(tick.bid-first_bid)/_Point;
//--- выведем сообщение, чтобы отслеживать работу советника
   string comm="С момента запуска:\r\n\x25CF прошло секунд: "+
               IntegerToString(tick.time-launch_time)+" ;"+
               "\r\n\x25CF поступило тиков: "+(string)tick_counter+" ;"+
               "\r\n\x25CF цена прошла в пунктах: "+StringFormat("%G",distance);
   Comment(comm);
//--- секция проверки условий для закрытия терминала
   if(tick_counter>=tiks_before)
      TerminalClose(0);    // выход по счетчику тиков
   if(distance>pips_to_go)
      TerminalClose(1);    // прошли вверх на pips_to_go пипсов
   if(distance<-pips_to_go)
      TerminalClose(-1);   // прошли вниз на pips_to_go пипсов
   if(tick.time-launch_time>seconds_st)
      TerminalClose(100);  // завершение работы по таймауту
//---
  }
```

Смотри также

Выполнение программ, Ошибки выполнения, Причины деинициализации

© 2000-2025, MetaQuotes Ltd.

TesterHideIndicators

Задает режим показа/сокрытия индикаторов, которые используются в эксперте. Функция предназначена для управления видимостью используемых индикаторов только при тестировании.

```
void  TesterHideIndicators(
   bool      hide     // флаг
   );
```

Параметры

hide

[in] Флаг сокрытия индикаторов при тестировании. Укажите true, если необходимо скрывать создаваемые индикаторы, иначе false.

Возвращаемое значение

Нет.

Примечание

По умолчанию на графике визуального тестирования показываются все индикаторы, которые создаются в тестируемом эксперте. Также эти индикаторы показываются на графике, который автоматически открывается по окончании тестирования. Функция TesterHideIndicators() позволяет разработчику в коде эксперта прописать запрет на показ используемых индикаторов.

Для того чтобы запретить показ используемого индикатора при тестировании советника, необходимо перед созданием его хендла вызвать TesterHideIndicators() с параметром true - все создаваемые после этого индикаторы будут помечены флагом скрытия. Индикаторы, помеченные флагом скрытия, не показываются при визуальном тестировании и на графике, который автоматически открывается при завершении тестирования.

Чтобы отключить режим скрытия вновь создаваемых индикаторов, нужно вновь вызвать TesterHideIndicators(), но уже с параметром false. На график тестирования могут быть выведены только те индикаторы, которые непосредственно создаются из тестируемого эксперта. Это правило относится только к тем случаям, когда нет ни одного шаблона в папке <каталог_данных>MQL5\Profiles\Templates.

Если в папке <каталог_данных>MQL5\Profiles\Templates присутствует специальный шаблон <имя_эксперта>.tpl, то при визуальном тестировании и на графике тестирования будут показаны только индикаторы из данного шаблона. В этом случае никакие индикаторы, используемые в тестируемом эксперте, показаны не будут. Даже если в коде советника вызывалась функция TesterHideIndicators() с параметром true.

Если же в папке <каталог_данных>MQL5\Profiles\Templates нет специального шаблона <имя_эксперта>.tpl, но есть шаблон tester.tpl, то при визуальном тестировании и на графике тестирования будут показаны индикаторы из шаблона tester.tpl и индикаторы из советника, которые не запрещены к показу с помощью функции TesterHideIndicators(). Если шаблона tester.tpl нет, то вместо него будут использованы индикаторы из шаблона default.tpl.

Если тестер стратегий не найдет ни одного подходящего шаблона (<имя_эксперта>.tpl, tester.tpl или default.tpl), то показ используемых в советнике индикаторов полностью управляется функцией TesterHideIndicators().

© 2000-2025, MetaQuotes Ltd.

Пример:

```
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_H
        {
         printf("Error creating EMA indicator");
         return(false);
        }
//--- succeed
   return(true);
  }
```

Смотри также

IndicatorRelease

© 2000-2025, MetaQuotes Ltd.

TesterStatistics

Возвращает значение указанного статистического показателя, рассчитанного по результатам тестирования

```
double  TesterStatistics(
   ENUM_STATISTICS statistic_id      // идентификатор
   );
```

Параметры

statistic_id

[in] Идентификатор статистического показателя из перечисления ENUM_STATISTICS.

Возвращаемое значение

Значение статистического показателя из результатов тестирования.

Примечание

Функция может быть вызвана внутри OnTester() или OnDeinit() в тестере. В других случаях результат неопределён.

Пример:

```
// Советник на основе стандартного файла "MACD Sample.mq5"
// показывает результат выполнения TesterStatistics() в обработчике события Tester
#define MACD_MAGIC 1234502
//---
#include <Trade\Trade.mqh>
#include <Trade\SymbolInfo.mqh>
#include <Trade\PositionInfo.mqh>
#include <Trade\AccountInfo.mqh>
//---
input double InpLots          =0.1; // Lots
input int    InpTakeProfit    =50;  // Take Profit (in pips)
input int    InpTrailingStop  =30;  // Trailing Stop Level (in pips)
input int    InpMACDOpenLevel =3;   // MACD open level (in pips)
input int    InpMACDCloseLevel=2;   // MACD close level (in pips)
input int    InpMATrendPeriod =26;  // MA trend period
//---
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- создадим все необходимые объекты
   if(!ExtExpert.Init())
      return(INIT_FAILED);
//--- успешная инициализация
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert new tick handling function                                |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   static datetime limit_time=0; // последнее время вызова с учетом timeout
//--- если время превысило установленное значение limit_time
   if(TimeCurrent()>=limit_time)
     {
      //--- проверим данные
      if(Bars(Symbol(),Period())>2*InpMATrendPeriod)
        {
         //--- в случае успешного выполнения увеличим limit_time на timeout секунд
         if(ExtExpert.Processing())
            limit_time=TimeCurrent()+ExtTimeOut;
        }
     }
  }
//+------------------------------------------------------------------+
//| Expert tester handling function                                  |
//+------------------------------------------------------------------+
double OnTester(void)
  {
   double ret=TesterStatistics(STAT_PROFIT_FACTOR);
   double profit=TesterStatistics(STAT_PROFIT);
   int    trades_total=(int)TesterStatistics(STAT_TRADES);
   int    profit_total=(int)TesterStatistics(STAT_PROFIT_TRADES);
   int    loss_total=(int)TesterStatistics(STAT_LOSS_TRADES);
   PrintFormat("%s: Profit = %.2f, trades total: %lu, profit trades total: %lu, loss t
   return(ret);
   /*
   Результат:
   OnTester: Profit = 209.84, trades total: 13, profit trades total: 11, loss trades t
   final balance 10209.84 USD
   OnTester result 3.020606644198363
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TesterStop

Отдает команду на завершении работы программы при тестировании.

```
void  TesterStop();
```

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Функция TesterStop() предназначена для штатного досрочного завершения работы советника на агенте тестирования - например, при достижении заданного количества убыточных сделок или заданного уровня просадки.

Вызов TesterStop() считается нормальным завершением тестирования, и поэтому будет вызвана функция OnTester() с отдачей тестеру стратегий всей накопленной торговой статистики и значения критерия оптимизации.

Вызов ExpertRemove() в тестере стратегий также означает нормальное завершение тестирования и позволяет получить торговую статистику, но при этом советник выгружается из памяти агента. В этом случае для выполнения прохода на следующем наборе параметров понадобится время на повторную загрузку программы. Поэтому для досрочного штатного завершения тестирования использование TesterStop() является наиболее предпочтительным вариантом.

Пример:

```
//--- defines
#define BALANCE_LOSS_STOP  100.0       // значение просадки баланса, при котором будет
#define EQUITY_LOSS_STOP   100.0       // значение просадки средств, при котором будет

//--- input parameters
input  double  InpLots        =  0.1;  // лоты
input  uint    InpStopLoss    =  50;   // Stop loss в пунктах
input  uint    InpTakeProfit  =  150;  // Take Profit  в пунктах
sinput ulong   InpMagic       =  123;  // MagicNumber
sinput ulong   InpDeviation   =  5;    // отклонение
//--- global variables
CTrade      trade;                     // экземпляр торгового класса
CSymbolInfo symb;                      // экземпляр класса символа
CAccountInfo account;                  // экземпляр класса торгового счета
...
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- Успешная инициализация
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- обновляем текущие котировки
   if(!symb.RefreshRates())
      return;
   ...

//--- если баланс или средства просели более, чем указано в макроподстановках BALANCE_
//--- значит считаем тест неудачным и вызываем функцию TesterStop()
//--- проверяем потерю баланса более, чем на BALANCE_LOSS_STOP
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()<balance_prev-BALANCE_LOSS_STOP)
        {
         PrintFormat("The initial balance of %.2f %s decreased by %.2f %s, and now has
         TesterStop();
         /*
         Результат:
         The initial balance of 10000.00 USD decreased by 100.10 USD, and now has a va
         TesterStop() called on 9% of testing interval
         */
        }
     }
//--- проверяем потерю средств более, чем на EQUITY_LOSS_STOP
   if(equity_prev!=account.Equity())
     {
      if(account.Equity()<equity_prev-EQUITY_LOSS_STOP)
        {
         PrintFormat("The initial equity of %.2f %s decreased by %.2f %s, and now has
         TesterStop();
         /*
         Результат:
         The initial equity of 10000.00 USD decreased by 100.10 USD, and now has a val
         TesterStop() called on 9% of testing interval
         */
        }
     }
  }
```

Смотри также

Выполнение программ, Тестирование торговых стратегий, ExpertRemove, SetReturnError

© 2000-2025, MetaQuotes Ltd.

TesterDeposit

Специальная функция для эмуляции операций внесения средств в процессе тестирования. Может быть использована в некоторых системах управления капиталом.

```
bool  TesterDeposit(
   double money      // размер вносимой суммы
   );
```

Параметры

money

[in] Размер денежных средств, которые необходимо внести на счет в валюте депозита.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
//--- defines
#define BALANCE_LOSS_DEPOSIT  100.0    // значение просадки баланса, при котором будут

//--- input parameters
input  double  InpLots        =  0.1;  // Lots
input  uint    InpStopLoss    =  50;   // Stop loss in points
input  uint    InpTakeProfit  =  150;  // Take Profit in points
sinput ulong   InpMagic       =  123;  // Magic number
sinput ulong   InpDeviation   =  5;    // Deviation
//--- global variables
CTrade      trade;                     // экземпляр торгового класса
CSymbolInfo symb;                      // экземпляр класса символа
CAccountInfo account;                  // экземпляр класса торгового счета
...
double      balance_dep_summ;          // общая сумма пополнений баланса
uint        balance_dep_total;         // количество пополнений баланса
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- сохраняем значения начального баланса
   balance_prev=account.Balance();
   balance_dep_summ=0;
   balance_dep_total=0;
//--- успешная инициализация
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- обновляем текущие котировки
   if(!symb.RefreshRates())
      return;
   ...

//--- если баланс просел более, чем указано в макроподстановке BALANCE_LOSS_DEPOSIT,
//--- значит считаем, что необходимо пополнить счёт, и вызываем функцию TesterDeposit
//--- проверяем потерю баланса более, чем на BALANCE_LOSS_DEPOSIT
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()<balance_prev-BALANCE_LOSS_DEPOSIT)
        {
         double loss=balance_prev-account.Balance();
         PrintFormat("The initial balance of %.2f %s decreased by %.2f %s. It is neces
         if(TesterDeposit(loss))
           {
            balance_dep_total++;
            balance_dep_summ+=loss;
            balance_prev=account.Balance();
            PrintFormat("Funds have been deposited into the account. Account balance:
            PrintFormat("Total deposits: %lu. Amount of deposits: %.2f %s.",balance_de
           }
         /*
         Результат:
         The initial balance of 10000.00 USD decreased by 116.00 USD. It is necessary
         deal #45 balance 116.00 [deposit] done
         Funds have been deposited into the account. Account balance: 10000.00 USD.
         Total deposits: 1. Amount of deposits: 116.00 USD.
         */
        }
     }
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- выходным значением обработчика укажем значение максимальной просадки баланса в д
   double ret=TesterStatistics(STAT_BALANCE_DD);
//--- выведем в журнал сообщение о просадке, о количестве пополнений и их общей сумме
   PrintFormat("%s: Maximum balance drawdown in money: %.2f %s. Total deposits: %lu. A
//--- возвращаем результат
   return(ret);
   /*
   Результат:
```

© 2000-2025, MetaQuotes Ltd.

```
   OnTester: Maximum balance drawdown in money: 5188.50 USD. Total deposits: 46. Amour
   final balance 4867.50 USD
   OnTester result 5188.5
   */
  }
```

Смотри также

TesterWithdrawal

© 2000-2025, MetaQuotes Ltd.

TesterWithdrawal

Специальная функция для эмуляции операций снятия средств в процессе тестирования. Может быть использована в некоторых системах управления капиталом.

```
bool  TesterWithdrawal(
   double money      // размер снимаемой суммы
   );
```

Параметры

money

[in] Размер денежных средств, которые необходимо снять со счета (в валюте депозита).

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
//--- defines
#define BALANCE_PROFIT_WITHDRAWAL   5  // значение прибыли на балансе, при которой бу

//--- input parameters
input  double  InpLots        =  0.1;  // Lots
input  uint    InpStopLoss    =  50;   // Stop loss in points
input  uint    InpTakeProfit  =  150;  // Take Profit in points
sinput ulong   InpMagic       =  123;  // Magic number
sinput ulong   InpDeviation   =  5;    // Deviation
//--- global variables
CTrade      trade;                     // экземпляр торгового класса
CSymbolInfo symb;                      // экземпляр класса символа
CAccountInfo account;                  // экземпляр класса торгового счета
...
double      balance_op_sum;            // общая сумма балансовых операций
uint        balance_op_total;          // количество балансовых операций
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- сохраняем значения начального баланса
   balance_prev=account.Balance();
   balance_op_sum=0;
   balance_op_total=0;
//--- успешная инициализация
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- обновляем текущие котировки
   if(!symb.RefreshRates())
      return;
   ...

//--- если прибыль на балансе больше текущего баланса на значение, указанное в макропо
//--- значит считаем, что необходимо снять эти средства со счёта, и вызываем функцию T
//--- Проверяем прибыль на балансе более, чем на BALANCE_PROFIT_WITHDRAWAL
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()>balance_prev+BALANCE_PROFIT_WITHDRAWAL)
        {
         double profit=account.Balance()-balance_prev;
         PrintFormat("The account balance has been increased by %.2f %s. Need to witho
         if(TesterWithdrawal(profit))
           {
            balance_op_total++;
            balance_op_summ+=profit;
            balance_prev=account.Balance();
            PrintFormat("Funds have been withdrawn from the account. Account balance:
            PrintFormat("Total withdrawals: %lu. Amount of withdrawals: %.2f %s.",bala
           }
         /*
         Результат:
         The account balance has been increased by 21.00 USD. Need to withdraw these f
         deal #13 balance -21.00 [withdrawal] done
         Funds have been withdrawn from the account. Account balance: 10000.00 USD.
         Total withdrawals: 1. Amount of withdrawals: 21.00 USD.
         */
        }
     }
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- выходным значением обработчика укажем значение максимальной просадки баланса в д
   double ret=TesterStatistics(STAT_BALANCE_DD);
//--- выведем в журнал сообщение о просадке, о количестве снятий и их общей сумме
   PrintFormat("%s: Maximum balance drawdown in money: %.2f %s. Total withdrawals: %lu
//--- возвращаем результат
   return(ret);
   /*
   Результат:
```

© 2000-2025, MetaQuotes Ltd.

```
   OnTester: Maximum balance drawdown in money: 5188.50 USD. Total withdrawals: 2. Amo
   final balance 4867.50 USD
   OnTester result 5188.5
   */
  }
```

Смотри также

TesterWithdrawal

© 2000-2025, MetaQuotes Ltd.

TranslateKey

Возвращает Unicode-символ по виртуальному коду клавиши, учитывая текущий язык ввода и состояние управляющих клавиш.

```
short  TranslateKey(
   int  key_code      // код клавиши для получения Unicode-символа
   );
```

Параметры

key_code

[in] Код клавиши.

Возвращаемое значение

Юникодный символ в случае успешного преобразования. В случае ошибки функция вернёт -1.

Примечание

Функция использует ToUnicodeEx для преобразования нажатых пользователем клавиш в Unicode-символы. Ошибка может возникнуть в том случае, если не сработает ToUnicodeEx - например, при попытке получить символ для клавиши SHIFT.

Пример:

```
void OnChartEvent(const int id,const long& lparam,const double& dparam,const string& s
  {
   if(id==CHARTEVENT_KEYDOWN)
     {
      short sym=TranslateKey((int)lparam);
      //--- если введённый символ успешно преобразован в Юникод
      if(sym>0)
         Print(sym,"'",ShortToString(sym),"'");
      else
         Print("Error in TranslateKey for key=",lparam);
     }
  }
```

Смотри также

События клиентского терминала, OnChartEvent

© 2000-2025, MetaQuotes Ltd.

ZeroMemory

Функция обнуляет переменную, переданную ей по ссылке.

```
void  ZeroMemory(
   void & variable      // обнуляемая переменная
   );
```

Параметры

variable

[in] [out] Переменная, передаваемая по ссылке, которую требуется обнулить (инициализировать нулевыми значениями).

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Если параметром функции является строка, то данный вызов будет эквивалентен указанию для нее значения NULL.
Для простых типов и их массивов, а также структур/классов, состоящих из таких типов, это простое обнуление.
Для объектов, содержащих строки и динамические массивы, производится вызов ZeroMemory() для каждого члена.
Для любых массивов, не защищенных модификатором const, производится обнуление всех элементов.
Для массивов сложных объектов происходит вызов ZeroMemory() для каждого элемента.

Функция ZeroMemory() не применима для классов с защищенными членами или наследованием.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявляем и инициализируем строку
   string str="Test ZeroMemory func";
//--- выведем в журнал строку до применения к ней ZeroMemory()
   PrintFormat("The line before applying ZeroMemory() to it: '%s'",str);
//--- обнулим строку и выведем в журнал результат
   ZeroMemory(str);
   Print("The same line after applying ZeroMemory() to it: '",str,"'");
/*
   Результат:
   The line before applying ZeroMemory() to it: 'Test ZeroMemory func'
   The same line after applying ZeroMemory() to it: ''
*/
```

© 2000-2025, MetaQuotes Ltd.

```
//--- объявляем и инициализируем переменную с типом int
   int var=123;
//--- выведем в журнал строку до применения к ней ZeroMemory()
   PrintFormat("\nThe integer variable before applying ZeroMemory() to it: %d",var);
//--- обнулим переменную и выведем в журнал результат
   ZeroMemory(var);
   PrintFormat("The same variable after applying ZeroMemory() to it: %d",var);
/*
   Результат:
   The integer variable before applying ZeroMemory() to it: 123
   The same variable after applying ZeroMemory() to it: 0
*/

//--- объявляем и инициализируем массив с типом int
   int arr[]={0,1,2,3,4,5,6,7,8,9};
//--- выведем в журнал массив до применения к нему ZeroMemory()
   Print("\nThe integer array before applying ZeroMemory() to it:");
   ArrayPrint(arr);
//--- обнулим массив и выведем в журнал результат
   ZeroMemory(arr);
   Print("The same array after applying ZeroMemory() to it:");
   ArrayPrint(arr);
/*
   Результат:
   The integer array before applying ZeroMemory() to it:
   0 1 2 3 4 5 6 7 8 9
   The same array after applying ZeroMemory() to it:
   0 0 0 0 0 0 0 0 0 0
*/

//--- объявляем структуру из двух полей - строкового и целочисленного
   struct STest
     {
      string   var_string;
      long     var_long;
     };
//--- объявляем и инициализируем массив с типом структуры STest
   STest arr_struct[]={ {"0",0}, {"1",1}, {"2",2}, {"3",3} };
//--- выведем в журнал массив структур до применения к нему ZeroMemory()
   Print("\nThe array struct before applying ZeroMemory() to it:");
   ArrayPrint(arr_struct);
//--- обнулим массив структур и выведем в журнал результат
   ZeroMemory(arr_struct);
   Print("The same array struct after applying ZeroMemory() to it:");
   ArrayPrint(arr_struct);
/*
   Результат:
   The array struct before applying ZeroMemory() to it:
```

© 2000-2025, MetaQuotes Ltd.

```
        [var_string] [var_long]
   [0] "0"                   0
   [1] "1"                   1
   [2] "2"                   2
   [3] "3"                   3
   The same array struct after applying ZeroMemory() to it:
        [var_string] [var_long]
   [0] null                  0
   [1] null                  0
   [2] null                  0
   [3] null                  0
*/
  }
```

© 2000-2025, MetaQuotes Ltd.

Группа функций для работы с массивами

Допускаются не более чем четырехмерные массивы. Индексация каждого измерения производится от 0 до *размер_измерения-1*. В частном случае одномерного массива из 50 элементов обращение к первому элементу будет выглядеть как array[0], к последнему элементу - array[49].

Функция	Действие
ArrayBsearch	Возвращает индекс первого найденного элемента в первом измерении массива
ArrayCopy	Копирует один массив в другой
ArrayCompare	Возвращает результат сравнения двух массивов простых типов или пользовательских структур, не имеющих сложных объектов
ArrayFree	Освобождает буфер любого динамического массива и устанавливает размер нулевого измерения в 0 (ноль)
ArrayGetAsSeries	Проверяет направление индексации массива
ArrayInitialize	Устанавливает все элементы числового массива в одну величину
ArrayFill	Заполняет числовой массив указанным значением
ArrayIsSeries	Проверяет, является ли массив таймсерией
ArrayIsDynamic	Проверяет, является ли массив динамическим
ArrayMaximum	Поиск элемента с максимальным значением
ArrayMinimum	Поиск элемента с минимальным значением
ArrayPrint	Выводит в журнал массив простого типа или простой структуры
ArrayRange	Возвращает число элементов в указанном измерении массива
ArrayResize	Устанавливает новый размер в первом измерении массива
ArrayInsert	Вставляет в массив-приемник из массива-источника указанное число элементов
ArrayRemove	Удаляет из массива указанное число элементов начиная с указанного индекса
ArrayReverse	Разворачивает в массиве указанное число элементов начиная с указанного индекса
ArraySetAsSeries	Устанавливает направление индексирования в массиве
ArraySize	Возвращает количество элементов в массиве
ArraySort	Сортировка числовых массивов по первому измерению
ArraySwap	Обменивает между собой содержимое двух динамических массивов одного типа

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
ArrayToFP16	Производит копирование массива типа float или double в массив типа ushort с заданным форматом
ArrayToFP8	Производит копирование массива типа float или double в массив типа uchar с заданным форматом
ArrayFromFP16	Производит копирование массива типа ushort в массив типа float или double с заданным форматом
ArrayFromFP8	Производит копирование массива типа uchar в массив типа float или double с заданным форматом

© 2000-2025, MetaQuotes Ltd.

ArrayBsearch

Ищет указанное значение в отсортированном по возрастанию многомерном числовом массиве. Поиск производится по элементам первого измерения.

Для поиска в массиве типа double

```
int  ArrayBsearch(
   const double&    array[],   // массив для поиска
   double           value      // что ищем
   );
```

Для поиска в массиве типа float

```
int  ArrayBsearch(
   const float&    array[],   // массив для поиска
   float           value      // что ищем
   );
```

Для поиска в массиве типа long

```
int  ArrayBsearch(
   const long&    array[],   // массив для поиска
   long           value      // что ищем
   );
```

Для поиска в массиве типа int

```
int  ArrayBsearch(
   const int&    array[],   // массив для поиска
   int           value      // что ищем
   );
```

Для поиска в массиве типа short

```
int  ArrayBsearch(
   const short&    array[],   // массив для поиска
   short           value      // что ищем
   );
```

Для поиска в массиве типа char

```
int  ArrayBsearch(
   const char&    array[],   // массив для поиска
   char           value      // что ищем
   );
```

Параметры

array[]

[in] Числовой массив для поиска.

value

© 2000-2025, MetaQuotes Ltd.

[in] Значение для поиска.

Возвращаемое значение

Возвращает индекс найденного элемента. Если искомое значение не найдено, то возвращает индекс ближайшего по значению элемента.

Примечание

Двоичный поиск обрабатывает только сортированные массивы. Для сортировки числового массива используется функция ArraySort().

Пример:

```
#property description "Скрипт на основе данных индикатора RSI выводит в окне графика"
#property description "данные о том, как часто рынок находился в зонах перекупленности
#property description "и перепроданности в указанном промежутке времени."
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input int                InpMAPeriod=14;                  // Период скользящей средн
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;     // Тип цены
input double             InpOversoldValue=30.0;           // Уровень перепроданности
input double             InpOverboughtValue=70.0;         // Уровень перекупленности
input datetime           InpDateStart=D'2012.01.01 00:00'; // Дата начала анализа
input datetime           InpDateFinish=D'2013.01.01 00:00'; // Дата конца анализа
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double rsi_buff[]; // массив значений индикатора
   int    size=0;     // размер массива
//--- получим хэндл индикатора RSI
   ResetLastError();
   int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- не удалось получить хэндл индикатора
      PrintFormat("Ошибка получения хэндла индикатора. Код ошибки = %d",GetLastError()
      return;
     }
//--- находимся в цикле, пока индикатор не рассчитает все свои значения
   while(BarsCalculated(rsi_handle)==-1)
     {
      //--- выходим, если пользователь принудительно завершил работу скрипта
      if(IsStopped())
         return;
      //--- задержка, чтобы индикатор успел вычислить свои значения
      Sleep(10);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- скопируем значения индикатора за определенный период
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения индикатора. Код ошибки = %d",GetLas
      return;
     }
//--- получим размер массива
   size=ArraySize(rsi_buff);
//--- отсортируем массив
   ArraySort(rsi_buff);
//--- узнаем какой процент времени рынок находился в зоне перепроданности
   double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- узнаем какой процент времени рынок находился в зоне перекупленности
   double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)siz
//--- сформируем строки для вывода данных
   string str="С "+TimeToString(InpDateStart,TIME_DATE)+" по "
              +TimeToString(InpDateFinish,TIME_DATE)+" рынок находился:";
   string str_ovb="в зоне перекупленности "+DoubleToString(ovb,2)+"% времени";
   string str_ovs="в зоне перепроданности "+DoubleToString(ovs,2)+"% времени";
//--- отобразим данные на графике
   CreateLabel("top",5,60,str,clrDodgerBlue);
   CreateLabel("overbought",5,35,str_ovb,clrDodgerBlue);
   CreateLabel("oversold",5,10,str_ovs,clrDodgerBlue);
//--- перерисуем график
   ChartRedraw(0);
//--- задержка
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Вывод комментария в левый нижний угол графика                    |
//+------------------------------------------------------------------+
void CreateLabel(const string name,const int x,const int y,
                 const string str,const color clr)
  {
//--- создание метки
   ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- привязка метки к левому нижнему углу
   ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- изменим положение точки привязки
   ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- расстояние по оси X от точки привязки
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- расстояние по оси Y от точки привязки
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
//--- текст метки
   ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- цвет текста
   ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- размер текста
   ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayCopy

Производит копирование одного массива в другой.

```
int  ArrayCopy(
   void&        dst_array[],          // куда копируем
   const void&  src_array[],          // откуда копируем
   int          dst_start=0,          // с какого индекса пишем в приемник
   int          src_start=0,          // с какого индекса копируем из источника
   int          count=WHOLE_ARRAY     // сколько элементов
   );
```

Параметры

dst_array[]

[out] Массив-приемник.

src_array[]

[in] Массив-источник.

dst_start=0

[in] Начальный индекс для приемного массива. По умолчанию, стартовый индекс - 0.

src_start=0

[in] Начальный индекс для исходного массива. По умолчанию, стартовый индекс - 0.

count=WHOLE_ARRAY

[in] Количество элементов, которые нужно скопировать. По умолчанию копируется весь массив (count=WHOLE_ARRAY).

Возвращаемое значение

Возвращает количество скопированных элементов.

Примечание

Если count<0 либо count>src_size-src_start, то копируется весь остаток массива. Массивы копируются слева направо. Для серийных массивов правильно переопределяется стартовая позиция с учетом копирования слева направо.

Если массивы разных типов, то при копировании производится попытка преобразования каждого элемента исходного массива к типу приемного массива. Строковый массив можно скопировать только в строковый массив. Массивы классов и структур, содержащих объекты, требующие инициализации, не копируются. Массив структур можно скопировать только в массив того же самого типа.

Для динамических массивов с индексацией как в таймсериях производится автоматическое увеличение размера массива-приемника до количества копируемых данных (в случае, если количество копируемых данных превышает его размер). Автоматическое уменьшение размера массива-приемника не производится.

Пример:

```
#property description "Индикатор выделяет цветом свечи, которые являются локальными"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "максимумами и минимумами. Длину интервала для нахождения"
#property description "экстремумов можно задать при помощи входного параметра."
//--- настройки индикатора
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot
#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
//--- предопределенная константа
#define INDICATOR_EMPTY_VALUE 0.0
//--- входные параметры
input int InpNum=4; // Длина полуинтервала
//--- индикаторные буферы
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- глобальные переменные
int    ExtStart=0; // индекс первой свечи, которая не является экстремумом
int    ExtCount=0; // количество свечей не экстремумов в данном интервале
//+------------------------------------------------------------------+
//| Закрашивание свечей не экстремумов                               |
//+------------------------------------------------------------------+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
  {
//--- закрашиваем свечи
   ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ExtOpen);
   SetIndexBuffer(1,ExtHigh);
   SetIndexBuffer(2,ExtLow);
   SetIndexBuffer(3,ExtClose);
   SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
//--- зададим значение, которое не будет отображаться
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- зададим имена индикаторных буферов для отображения в окне данных
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- установим прямое направление индексации в таймсериях
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- переменная начала для расчета баров
   int start=prev_calculated;
//--- для первых InpNum*2 баров расчет не проводим
   if(start==0)
     {
      start+=InpNum*2;
      ExtStart=0;
      ExtCount=0;
     }
//--- если только что сформировался бар, то проверим следующий потенциальный экстремум
   if(rates_total-start==1)
      start--;
//--- индекс бара, который будем проверять на экстремум
   int ext;
//--- цикл расчета значений индикатора
   for(int i=start;i<rates_total-1;i++)
     {
      //--- изначально на i-ом баре без отрисовки
      ExtOpen[i]=0;
      ExtHigh[i]=0;
      ExtLow[i]=0;
      ExtClose[i]=0;
      //--- индекс экстремума для проверки
      ext=i-InpNum;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- проверка на локальный максимум
      if(IsMax(high,ext))
        {
         //--- пометим свечу экстремум
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=1;
         //--- остальные свечи до экстремума пометим нейтральным цветом
         FillCandles(open,high,low,close);
         //--- изменяем значения переменных
         ExtStart=ext+1;
         ExtCount=0;
         //--- переходим к следующей итерации
         continue;
        }
      //--- проверка на локальный минимум
      if(IsMin(low,ext))
        {
         //--- пометим свечу экстремум
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=2;
         //--- остальные свечи до экстремума пометим нейтральным цветом
         FillCandles(open,high,low,close);
         //--- изменяем значения переменных
         ExtStart=ext+1;
         ExtCount=0;
         //--- переходим к следующей итерации
         continue;
        }
      //--- увеличиваем количество не экстремумов в данном интервале
      ExtCount++;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Является ли текущий элемент массива локальным максимумом         |
//+------------------------------------------------------------------+
bool IsMax(const double &price[],const int ind)
  {
//--- переменная начала интервала
   int i=ind-InpNum;
//--- переменная окончания интервала
   int finish=ind+InpNum+1;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- проверка для первой половины интервала
   for(;i<ind;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- проверка для второй половины интервала
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- это экстремум
   return(true);
  }
//+------------------------------------------------------------------+
//| Является ли текущий элемент массива локальным минимумом          |
//+------------------------------------------------------------------+
bool IsMin(const double &price[],const int ind)
  {
//--- переменная начала интервала
   int i=ind-InpNum;
//--- переменная окончания интервала
   int finish=ind+InpNum+1;
//--- проверка для первой половины интервала
   for(;i<ind;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- проверка для второй половины интервала
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- это экстремум
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayCompare

Возвращает результат сравнения двух массивов одинакового типа. Может использоваться для сравнения массивов простых типов или пользовательских структур, не имеющих сложных объектов - то есть не содержащих строк, динамических массивов, классов или других структур содержащих сложные объекты.

```
int  ArrayCompare(
   const void&  array1[],           // первый массив
   const void&  array2[],           // второй массив
   int          start1=0,           // начальное смещение в первом массиве
   int          start2=0,           // начальное смещение во втором массиве
   int          count=WHOLE_ARRAY   // количество элементов для сравнения
   );
```

Параметры

array1[]

[in] Первый массив.

array2[]

[in] Второй массив.

start1=0

[in] Начальный индекс элемента в первом массиве, с которого начнется сравнение. По умолчанию стартовый индекс - 0.

start2=0

[in] Начальный индекс элемента во втором массиве, с которого начнется сравнение. По умолчанию стартовый индекс - 0.

count=WHOLE_ARRAY

[in] Количество элементов, которые нужно сравнить. По умолчанию в сравнении участвуют все элементы обоих массивов (count=WHOLE_ARRAY).

Возвращаемое значение

- -1, если array1[] меньше array2[]
- 0, если array1[] и array2[] равны
- 1, если array1[] больше array2[]
- -2, при возникновении ошибки из-за несовместимости типов сравниваемых массивов, или при значениях start1, start2 или count, приводящих к выходу за пределы массива.

Примечание

При разных размерах сравниваемых массивов и с указанным значением count=WHOLE_ARRAY для случая, когда один массив является точным подмножеством другого, функция не вернёт 0 (массивы не будут считаться равными). В этом случае будет возвращен результат сравнения размеров этих массивов: -1, если размер array1[] меньше размера array2[] или 1 в противном случае.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
//--- глобальные переменные
double   ExtArrayFirst[];
double   ExtArraySecond[];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- устанавливаем размеры массивам
   if(ArrayResize(ExtArrayFirst,10)!=10)
     {
      Print("ArrayResize() failed for ExtArrayFirst. Error code: ",GetLastError());
      return;
     }
   if(ArrayResize(ExtArraySecond,10)!=10)
     {
      Print("ArrayResize() failed for ExtArraySecond. Error code: ",GetLastError());
      return;
     }

//--- в цикле заполняем массивы значениями индексов i и j
   int total=ArraySize(ExtArrayFirst);
   for(int i=0, j=total-1; i<total; i++,j--)
     {
      //--- массив ExtArrayFirst заполняем слева-направо
      //--- массив ExtArraySecond заполняем справа-налево
      ExtArrayFirst[i]=i;
      ExtArraySecond[i]=j;
     }
//--- сравним массивы и распечатаем в журнале результат
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
   Результат:
   ExtArrayFirst:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   ExtArraySecond:
   9.00000 8.00000 7.00000 6.00000 5.00000 4.00000 3.00000 2.00000 1.00000 0.00000
   Result ArrayCompare(): ExtArrayFirst is smaller than ExtArraySecond (result = -1)
   */

//--- теперь перевернём массивы
//--- в цикле заполняем массивы значениями индексов i и j
   for(int i=0, j=total-1; i<total; i++,j--)
     {
      //--- массив ExtArrayFirst заполняем справа-налево
      //--- массив ExtArraySecond заполняем слева-направо
      ExtArrayFirst[i]=j;
      ExtArraySecond[i]=i;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- сравним массивы и распечатаем в журнале результат
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
   Результат:
   ExtArrayFirst:
   9.00000 8.00000 7.00000 6.00000 5.00000 4.00000 3.00000 2.00000 1.00000 0.00000
   ExtArraySecond:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   Result ArrayCompare(): ExtArrayFirst is larger than ExtArraySecond (result = 1)
   */

//--- теперь заполним массивы в одном направлении
//--- в цикле заполняем массивы значениями индекса i
   for(int i=0; i<total; i++)
     {
      //--- оба массива заполняем слева-направо
      ExtArrayFirst[i]=i;
      ExtArraySecond[i]=i;
     }
//--- сравним массивы и распечатаем в журнале результат
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
   Результат:
   ExtArrayFirst:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   ExtArraySecond:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   Result ArrayCompare(): ExtArrayFirst and ExtArraySecond are equal (result = 0)
   */
  }
//+------------------------------------------------------------------+
//| Сравнение и вывод результата                                     |
//+------------------------------------------------------------------+
void ArrayComparePrint(const double &array1[], const double &array2[])
  {
   //--- распечатаем заголовок и содержимое массивов
   Print("ExtArrayFirst:");
   ArrayPrint(array1);
   Print("ExtArraySecond:");
   ArrayPrint(array2);
   //--- сравним массивы и распечатаем результат сравнения
   int    res=ArrayCompare(array1,array2);
   string res_str=(res>0 ? "ExtArrayFirst is larger than ExtArraySecond" : res<0 ? "Ex
   PrintFormat("Result ArrayCompare(): %s (result = %d)\n",res_str,res);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

ArrayFree

Освобождает буфер любого динамического массива и устанавливает размер нулевого измерения в 0.

```
void  ArrayFree(
   void&  array[]      // массив
   );
```

Параметры

array[]

[in] Динамический массив.

Возвращаемое значение

Нет возвращаемого значения.

Примечание

При написании скриптов и индикаторов необходимость в использовании функции ArrayFree() может возникнуть не часто: так как при завершении работы скрипта вся использованная память сразу же освобождается, а в пользовательских индикаторах основная работа с массивами представляет собою доступ к индикаторным буферам, размеры которых автоматически управляются исполняющей подсистемой терминала.

Если в программе необходимо самостоятельно управлять памятью в сложных динамических условиях, то функция ArrayFree() позволит явным образом и немедленно освобождать память, занятую ненужным уже динамическим массивом.

Пример:

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
//--- предопределенные константы
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+------------------------------------------------------------------+
//| Класс диалога для работы с памятью                               |
//+------------------------------------------------------------------+
class CMemoryControl : public CAppDialog
  {
private:
   //--- размер массива
   int               m_arr_size;
   //--- массивы
   char              m_arr_char[];
   int               m_arr_int[];
```

© 2000-2025, MetaQuotes Ltd.

```
   float             m_arr_float[];
   double            m_arr_double[];
   long              m_arr_long[];
   //--- надписи
   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;
   //--- кнопки
   CButton           m_button_add;
   CButton           m_button_free;
   //--- списки
   CComboBox         m_combo_box_step;
   CComboBox         m_combo_box_type;
   //--- текущее значение типа массива из списка
   int               m_combo_box_type_value;

public:
                     CMemoryControl(void);
                    ~CMemoryControl(void);
   //--- метод создания объекта класса
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- обработчик событий графика
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- создание надписи
   bool              CreateLabel(CLabel &lbl,const string name,const int x,const int y
   //--- создание элементов управления
   bool              CreateButton(CButton &button,const string name,const int x,const
   bool              CreateComboBoxStep(void);
   bool              CreateComboBoxType(void);
   //--- обработчики событий
   void              OnClickButtonAdd(void);
   void              OnClickButtonFree(void);
   void              OnChangeComboBoxType(void);
   //--- методы работы с текущим массивом
   void              CurrentArrayFree(void);
   bool              CurrentArrayAdd(void);
  };
//+------------------------------------------------------------------+
//| Освобождение памяти текущего массива                             |
//+------------------------------------------------------------------+
void CMemoryControl::CurrentArrayFree(void)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- сброс размера массива
   m_arr_size=0;
//--- освобождение массива
   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);
  }
//+------------------------------------------------------------------+
//| Попытка добавления памяти для текущего массива                   |
//+------------------------------------------------------------------+
bool CMemoryControl::CurrentArrayAdd(void)
  {
//--- если размер используемой памяти больше чем размер физической памяти, то выходим
   if(TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMOR
      return(false);
//--- попытка выделения памяти в зависимости от текущего типа
   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);
//--- память выделена
   return(true);
  }
//+------------------------------------------------------------------+
//| Обработка событий                                                |
//+------------------------------------------------------------------+
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)
//+------------------------------------------------------------------+
//| Конструктор                                                      |
//+------------------------------------------------------------------+
CMemoryControl::CMemoryControl(void)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
  }
//+------------------------------------------------------------------+
//| Деструктор                                                       |
//+------------------------------------------------------------------+
CMemoryControl::~CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| Метод создания объекта класса                                    |
//+------------------------------------------------------------------+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                            const int x1,const int y1,const int x2,const int y2)
  {
//--- создание объекта базового класса
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- подготовка строк для надписей
   string str_physical="Memory physical = "+(string)TerminalInfoInteger(TERMINAL_MEMOR
   string str_total="Memory total = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_TOTA
   string str_available="Memory available = "+(string)TerminalInfoInteger(TERMINAL_MEM
   string str_used="Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_USED)+
//--- создание надписей
   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);
//--- создание элементов управления
   if(!CreateButton(m_button_add,"add_button",X_START+15,Y_START+245,"Add",12,clrBlue)
      return(false);
   if(!CreateButton(m_button_free,"free_button",X_START+75,Y_START+245,"Free",12,clrBl
      return(false);
   if(!CreateComboBoxType())
      return(false);
   if(!CreateComboBoxStep())
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- инициализация переменной
   m_arr_size=0;
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Создание кнопки                                                  |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                  const int y,const string str,const int font_size,
                                  const int clr)
  {
//--- создание кнопки
   if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
      return(false);
//--- текст
   if(!button.Text(str))
      return(false);
//--- размер шрифта
   if(!button.FontSize(font_size))
      return(false);
//--- цвет надписи
   if(!button.Color(clr))
      return(false);
//--- добавляем кнопку в элементы контроля
   if(!Add(button))
      return(false);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Создание списка для размера массива                              |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxStep(void)
  {
//--- создание списка
   if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- добавление элементов в список
   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);
//--- установим текущий элемент списка
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!m_combo_box_step.SelectByValue(1000000))
      return(false);
//--- добавляем список в элементы контроля
   if(!Add(m_combo_box_step))
      return(false);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Создание списка для типа массива                                 |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxType(void)
  {
//--- создание списка
   if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- добавление элементов в список
   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);
//--- установим текущий элемент списка
   if(!m_combo_box_type.SelectByValue(3))
      return(false);
//--- запомним текущий элемент списка
   m_combo_box_type_value=3;
//--- добавляем список в элементы контроля
   if(!Add(m_combo_box_type))
      return(false);
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Создание надписи                                                 |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                 const int y,const string str,const int font_size,
                                 const int clr)
  {
//--- создание надписи
   if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
      return(false);
//--- текст
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!lbl.Text(str))
      return(false);
//--- размер шрифта
   if(!lbl.FontSize(font_size))
      return(false);
//--- цвет
   if(!lbl.Color(clr))
      return(false);
//--- добавляем надпись в элементы контроля
   if(!Add(lbl))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Обработчик события нажатия на кнопку "Add"                       |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonAdd(void)
  {
//--- увеличим размер массива
   m_arr_size+=(int)m_combo_box_step.Value();
//--- пытаемся выделить память под текущий массив
   if(CurrentArrayAdd())
     {
      //--- память выделена, выведем текущее состояние на экран
      m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TE
      m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEM
      m_lbl_array_size.Text("Array size = "+IntegerToString(m_arr_size));
      m_lbl_error.Text("");
     }
   else
     {
      //--- не удалось выделить память, выводим сообщение об ошибке
      m_lbl_error.Text("Array is too large, error!");
      //--- вернем предыдущий размер массива
      m_arr_size-=(int)m_combo_box_step.Value();
     }
  }
//+------------------------------------------------------------------+
//| Обработчик события нажатия на кнопку "Free"                      |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonFree(void)
  {
//--- освобождаем память текущего массива
   CurrentArrayFree();
//--- выведем текущее состояние на экран
   m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TERMI
   m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMORY
   m_lbl_array_size.Text("Array size = 0");
```

© 2000-2025, MetaQuotes Ltd.

```
   m_lbl_error.Text("");
  }
//+------------------------------------------------------------------+
//| Обработчик события изменения списка                              |
//+------------------------------------------------------------------+
void CMemoryControl::OnChangeComboBoxType(void)
  {
//--- проверка, изменился ли тип массива
   if(m_combo_box_type.Value()!=m_combo_box_type_value)
     {
      //--- освобождаем память текущего массива
      OnClickButtonFree();
      //--- работаем с другим типом массива
      m_combo_box_type_value=(int)m_combo_box_type.Value();
      //--- выведем на экран новый тип массива
      if(m_combo_box_type_value==0)
         m_lbl_array_type.Text("Array type = char");
      if(m_combo_box_type_value==1)
         m_lbl_array_type.Text("Array type = int");
      if(m_combo_box_type_value==2)
         m_lbl_array_type.Text("Array type = float");
      if(m_combo_box_type_value==3)
         m_lbl_array_type.Text("Array type = double");
      if(m_combo_box_type_value==4)
         m_lbl_array_type.Text("Array type = long");
     }
  }
//--- объект класса CMemoryControl
CMemoryControl ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- создание диалога
   if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
      return(INIT_FAILED);
//--- запуск
   ExtDialog.Run();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ExtDialog.Destroy(reason);
```

© 2000-2025, 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayGetAsSeries

Проверяет направление индексации массива.

```
bool  ArrayGetAsSeries(
   const void&  array[]    // массив для проверки
   );
```

Параметры

array[]

[in] Проверяемый массив.

Возвращаемое значение

Возвращает true, если у указанного массива установлен флаг AS_SERIES, то есть доступ к массиву осуществляется задом наперед как в таймсерии. Таймсерия отличается от обычного массива тем, что индексация элементов таймсерии производится от конца массива к началу (от самых свежих данных к самым старым).

Примечание

Для проверки массива на принадлежность к таймсерии следует применять функцию ArrayIsSeries(). Массивы ценовых данных, переданных в качестве входных параметров в функцию OnCalculate(), не обязательно имеют направление индексации как у таймсерий. Нужное направление индексации можно установить функцией ArraySetAsSeries().

Пример:

```
#property description "Индикатор вычисляет абсолютные значения разницы между ценами"
#property description "Open и Close или High и Low, и отображает их в отдельном подокн
#property description "в виде гистограммы."
//--- настройки индикатора
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- входные параметры
input bool InpAsSeries=true; // Направление индексации в индикаторном буфере
input bool InpPrices=true;   // Цены для расчета (true - Open,Close; false - High,Low)
//--- индикаторный буфер
double ExtBuffer[];
//+------------------------------------------------------------------+
//| Вычисление значений индикатора                                   |
//+------------------------------------------------------------------+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double &buffer[])
  {
//--- переменная начала для рассчета баров
```

© 2000-2025, MetaQuotes Ltd.

```
   int start=prev_calculated;
//--- если значения индикатора уже были рассчитаны на предыдущем тике, то работаем на
   if(prev_calculated>0)
      start--;
//--- определим направление индексации в массивах
   bool as_series_first=ArrayGetAsSeries(first);
   bool as_series_second=ArrayGetAsSeries(second);
   bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- изменим направление индексации на прямое, если необходимо
   if(as_series_first)
      ArraySetAsSeries(first,false);
   if(as_series_second)
      ArraySetAsSeries(second,false);
   if(as_series_buffer)
      ArraySetAsSeries(buffer,false);
//--- рассчитаем значения индикатора
   for(int i=start;i<rates_total;i++)
      buffer[i]=MathAbs(first[i]-second[i]);
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка индикаторных буферов
   SetIndexBuffer(0,ExtBuffer);
//--- установим направление индексации в индикаторном буфере
   ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- проверяем для каких цен рассчитывается индикатор
   if(InpPrices)
     {
      //--- цены Open и Close
      PlotIndexSetString(0,PLOT_LABEL,"BodySize");
      //--- установим цвет индикатора
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
     }
   else
     {
      //--- цены High и Low
      PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
      //--- установим цвет индикатора
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

© 2000-2025, 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[])
  {
//--- расчет индикатора в зависимости от значения флага
   if(InpPrices)
      CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
   else
      CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Доступ к таймсериям, ArraySetAsSeries

© 2000-2025, MetaQuotes Ltd.

ArrayInitialize

Инициализирует числовой массив указанным значением.

Для инициализации массива типа char

```
int  ArrayInitialize(
   char     array[],      // инициализируемый массив
   char     value         // значение, которое будет установлено
   );
```

Для инициализации массива типа short

```
int  ArrayInitialize(
   short    array[],      // инициализируемый массив
   short    value         // значение, которое будет установлено
   );
```

Для инициализации массива типа int

```
int  ArrayInitialize(
   int      array[],      // инициализируемый массив
   int      value         // значение, которое будет установлено
   );
```

Для инициализации массива типа long

```
int  ArrayInitialize(
   long     array[],      // инициализируемый массив
   long     value         // значение, которое будет установлено
   );
```

Для инициализации массива типа float

```
int  ArrayInitialize(
   float    array[],      // инициализируемый массив
   float    value         // значение, которое будет установлено
   );
```

Для инициализации массива типа double

```
int  ArrayInitialize(
   double   array[],      // инициализируемый массив
   double   value         // значение, которое будет установлено
   );
```

Для инициализации массива типа bool

```
int  ArrayInitialize(
   bool     array[],      // инициализируемый массив
   bool     value         // значение, которое будет установлено
   );
```

© 2000-2025, MetaQuotes Ltd.

Для инициализации массива типа uint

```
int  ArrayInitialize(
   uint    array[],     // инициализируемый массив
   uint    value        // значение, которое будет установлено
   );
```

Параметры

array[]

[out] Числовой массив, который нужно инициализировать.

value

[in] Новое значение, которое нужно установить всем элементам массива.

Возвращаемое значение

Количество инициализированных элементов.

Примечание

Функция ArrayResize() позволяет задать для массива размер с некоторым запасом для его будущего увеличения без физического перераспределения памяти. Это сделано для улучшения быстродействия, так как операции по распределению памяти являются достаточно медленными.

Инициализация массива выражением ArrayInitialize(array, init_val) не означает инициализацию этим же значением и элементов резерва, выделенного для этого массива. При последующих увеличениях размера массива *array* функцией ArrayResize() в пределах текущего резерва, в конец массива добавляются элементы, значения которых не определены и, чаще всего, не равны *init_val*.

Пример:

```
void OnStart()
  {
//--- динамический массив
   double array[];
//--- зададим размер массива для 100 элементов и зарезервируем еще буфер в 10 элемент
   ArrayResize(array,100,10);
//--- инициализируем элементы массива значением EMPTY_VALUE=DBL_MAX
   ArrayInitialize(array,EMPTY_VALUE);
   Print("Значения последних 10 элементов массива после инициализации");
   for(int i=90;i<100;i++)
      printf("array[%d] = %G",i,array[i]);
//--- увеличим размер массива на 5 элементов
   ArrayResize(array,105);
   Print("Значения последних 10 элементов массива после ArrayResize(array,105)");
//--- значения последних 5 элементов были получены из буфера резерва
   for(int i=95;i<105;i++)
      printf("array[%d] = %G",i,array[i]);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayFill

Заполняет числовой массив указанным значением.

```
void  ArrayFill(
   void&  array[],        // массив
   int    start,          // индекс начального элемента
   int    count,          // количество элементов
   void   value           // значение, которым заполняется массив
   );
```

Параметры

array[]

[out] Массив простого типа (char, uchar, short, ushort, int, uint, long, ulong, bool, color, datetime, float, double).

start

[in] Индекс начального элемента (с какого элемента заполнять). При этом не учитывается установленный флаг серийности.

count

[in] Количество элементов, которое следует заполнить.

value

[in] Значение, которым заполняется массив.

Возвращаемое значение

Нет возвращаемого значения.

Примечание

При вызове функции ArrayFill() всегда подразумевается обычное направление индексации - слева направо, то есть изменение порядка доступа к элементам массива с помощью функции ArraySetAsSeries() не принимается во внимание.

Многомерный массив при обработке функцией ArrayFill() представляется одномерным, например, массив array[2][4] обрабатывается как array[8], поэтому при работе с этим массивом допустимо указать индекс начального элемента равным 5. Таким образом, вызов ArrayFill(array, 5, 2, 3.14) для массива array[2][4] заполнит значением 3.14 элементы массива array[1][1] и array[1][2].

Пример:

```
void OnStart()
  {
//--- объявляем динамический массив
   int a[];
//--- устанавливаем размер
   ArrayResize(a,10);
//--- заполняем начальные 5 элементов значением 123
   ArrayFill(a,0,5,123);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- заполняем 5 элементов (начиная с 5-го) значением 456
   ArrayFill(a,5,5,456);
//--- выводим значения всех элементов
   for(int i=0;i<ArraySize(a);i++)
      printf("a[%d] = %d",i,a[i]);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayIsDynamic

Проверяет, является ли массив динамическим.

```
bool  ArrayIsDynamic(
   const void&  array[]    // проверяемый массив
   );
```

Параметры

array[]

[in] Проверяемый массив.

Возвращаемое значение

Возвращает true, если указанный массив является динамическим, иначе возвращается false.

Пример:

```
#property description "Этот индикатор не вычисляет значений, а один раз пытается приме
#property description "вызов функции ArrayFree() к трем массивам: динамическому, стати
#property description "индикаторному буферу. Результаты выводятся в журнал экспертов."
//--- настройки индикатора
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- глобальные переменные
double ExtDynamic[];   // динамический массив
double ExtStatic[100]; // статический массив
bool   ExtFlag=true;   // флаг
double ExtBuff[];      // индикаторный буфер
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- выделим память для массива
   ArrayResize(ExtDynamic,100);
//--- indicator buffers mapping
   SetIndexBuffer(0,ExtBuff);
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- проведем анализ один раз
   if(ExtFlag)
     {
      //--- попробуем освободить память для массивов
      //--- 1. Динамический массив
      Print("+============================+");
      Print("1. Проверка динамического массива:");
      Print("Размер до освобождения памяти = ",ArraySize(ExtDynamic));
      Print("Это динамический массив = ",ArrayIsDynamic(ExtDynamic) ? "Да" : "Нет");
      //--- пытаемся освободить память массива
      ArrayFree(ExtDynamic);
      Print("Размер после освобождения памяти = ",ArraySize(ExtDynamic));
      //--- 2. Статический массив
      Print("2. Проверка статического массива:");
      Print("Размер до освобождения памяти = ",ArraySize(ExtStatic));
      Print("Это динамический массив = ",ArrayIsDynamic(ExtStatic) ? "Да" : "Нет");
      //--- пытаемся освободить память массива
      ArrayFree(ExtStatic);
      Print("Размер после освобождения памяти = ",ArraySize(ExtStatic));
      //--- 3. Индикаторный буфер
      Print("3. Проверка индикаторного буфера:");
      Print("Размер до освобождения памяти = ",ArraySize(ExtBuff));
      Print("Это динамический массив = ",ArrayIsDynamic(ExtBuff) ? "Да" : "Нет");
      //--- пытаемся освободить память массива
      ArrayFree(ExtBuff);
      Print("Размер после освобождения памяти = ",ArraySize(ExtBuff));
      //--- изменим значение флага
      ExtFlag=false;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Доступ к таймсериям и индикаторам

© 2000-2025, MetaQuotes Ltd.

ArrayIsSeries

Проверяет, является ли массив таймсерией.

```
bool  ArrayIsSeries(
   const void&  array[]    // проверяемый массив
   );
```

Параметры

array[]

[in] Проверяемый массив.

Возвращаемое значение

Возвращается true, если проверяемый массив является массивом-таймсерией, иначе возвращается false. Массивы, передаваемые в качестве параметра функции OnCalculate(), необходимо проверять на порядок доступа к элементам массива функцией ArrayGetAsSeries().

Пример:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(ArrayIsSeries(open))
      Print("open[] is timeseries");
   else
      Print("open[] is not timeseries!!!");
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Доступ к таймсериям и индикаторам

© 2000-2025, MetaQuotes Ltd.

ArrayMaximum

Ищет максимальный элемент в первом измерении многомерного числового массива.

```
int  ArrayMaximum(
   const void&   array[],              // массив для поиска
   int           start=0,              // с какого индекса начинаем поиск
   int           count=WHOLE_ARRAY     // количество проверяемых
   );
```

Параметры

array[]

[in] Числовой массив, в котором производится поиск.

start=0

[in] Начальный индекс для поиска.

count=WHOLE_ARRAY

[in] Количество элементов для поиска. По умолчанию, ищет во всем массиве (count=WHOLE_ARRAY).

Возвращаемое значение

Функция возвращает индекс найденного элемента с учетом серийности массива. В случае неудачи функция возвращает -1.

Примечание

Поиск максимального элемента производится с учетом значения флага AS_SERIES.

Функции ArrayMaximum и ArrayMinimum принимают в качестве параметра массив любой размерности, при этом поиск происходит только по первому (нулевому) измерению.

Пример:

```
#property description "Индикатор отображает свечи со старшего таймфрейма на текущем."
//--- настройки индикатора
#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
```

© 2000-2025, 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
//--- предопределенная константа
#define INDICATOR_EMPTY_VALUE 0.0
//--- входные параметры
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Таймфрейм для расчета индик
input datetime        InpDateStart=D'2013.01.01 00:00'; // Дата начала анализа
//--- индикаторные буферы для медвежьих свечей
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- индикаторные буферы для бычьих свечей
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- глобальные переменные
datetime ExtTimeBuff[];      // буфер для времени с высшего таймфрейма
int      ExtSize=0;          // размер буфера времени
int      ExtCount=0;         // индекс внутри буфера времени
int      ExtStartPos=0;      // начальная позиция для расчета индикатора
bool     ExtStartFlag=true;  // вспомогательный флаг для получения начальной позиции
datetime ExtCurrentTime[1];  // последнее время формирования бара со старшего таймфрей
datetime ExtLastTime;        // последнее время со старшего таймфрейма, для которого и
bool     ExtBearFlag=true;   // флаг для определения порядка записи данных в медвежьи
bool     ExtBullFlag=true;   // флаг для определения порядка записи данных в бычьи инд
int      ExtIndexMax=0;      // индекс максимального элемента в массиве
int      ExtIndexMin=0;      // индекс минимального элемента в массиве
int      ExtDirectionFlag=0; // направление движения цены у текущей свечи
//--- отступ между ценой открытия и закрытия свечи для правильной отрисовки
```

© 2000-2025, MetaQuotes Ltd.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//| Закрашивание основной части свечи                                |
//+------------------------------------------------------------------+
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)
  {
//--- найдем индексы максимального и минимального элементов в массивах
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // максимум в High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // минимум в Low
//--- определим сколько баров с текущего таймфрейма будем закрашивать
   int count=fill_index-start+1;
//--- если цена закрытия на первом баре больше цены закрытия на последнем - свеча медв
   if(open[start]>close[last])
     {
      //--- если до этого свеча была бычьей, то очищаем значения бычьих индикаторных б
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- медвежья свеча
      ExtDirectionFlag=-1;
      //--- формируем свечку
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- выход из функции
      return;
     }
//--- если цена закрытия на первом баре меньше цены закрытия на последнем - свеча быч
   if(open[start]<close[last])
     {
      //--- если до этого свеча была медвежьей, то очищаем значения медвежьих индикато
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- бычья свеча
      ExtDirectionFlag=1;
      //--- формируем свечку
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- выход из функции
      return;
     }
//--- если попали в эту часть функции, то значит цена открытия на первом баре равняет
//--- цене закрытия на последнем баре; будем считать такую свечу медвежьей
//--- если до этого свеча была бычьей, то очищаем значения бычьих индикаторных буферо
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- медвежья свеча
   ExtDirectionFlag=-1;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если цены закрытия и цены открытия равны, то используем сдвиг для корректного от
   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);
  }
//+------------------------------------------------------------------+
//| Закрашивание конца свечи                                         |
//+------------------------------------------------------------------+
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)
  {
//--- если всего один бар, то не рисуем
   if(last-start==0)
      return;
//--- если цена закрытия на первом баре больше цены закрытия на последнем - свеча медв
   if(open[start]>close[last])
     {
      //--- формируем конец свечи
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- выход из функции
      return;
     }
//--- если цена закрытия на первом баре меньше цены закрытия на последнем - свеча быч
   if(open[start]<close[last])
     {
      //--- формируем конец свечи
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- выход из функции
      return;
     }
//--- если попали в эту часть функции, то значит цена открытия на первом баре равняетс
//--- цене закрытия на последнем баре; будем считать такую свечу медвежьей
//--- формируем конец свечи
   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
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- проверка периода индикатора
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- отображение ценовых данных на переднем плане
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- привязка индикаторных буферов
   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);
//--- зададим некоторые значения свойств для построения индикатора
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // тип графического построен
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // стиль линии отрисовки
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // толщина линии отрисовки
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- если еще нет рассчитанных баров
   if(prev_calculated==0)
     {
      //--- получим время появления баров со старшего таймфрейма
      if(!GetTimeData())
         return(0);
     }
//--- установим прямое направление индексации
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- переменная начала для расчета баров
   int start=prev_calculated;
//--- если бар формируется, то пересчитываем значение индикатора на нем
   if(start!=0 && start==rates_total)
      start--;
//--- цикл расчета значений индикатора
   for(int i=start;i<rates_total;i++)
     {
      //--- заполняем i-ые элементы индикаторных буферов пустыми значениями
      FillIndicatorBuffers(i);
      //--- проводим вычисление для баров начиная с даты InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- в первый раз определим позицию с которой начнем отображать значения
         if(ExtStartFlag)
           {
            //--- запомним номер начального бара
            ExtStartPos=i;
            //--- определим первую дату со старшего таймфрейма, которая больше time[i]
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- изменим значение флага, чтобы больше не попадать в этот блок
            ExtStartFlag=false;
           }
         //--- проверка, есть ли еще в массиве элементы
         if(ExtCount<ExtSize)
           {
            //--- ждем, когда значение времени с текущего таймфрейма достигнет значени
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- рисуем главную часть свечи (без закрашивания между последним и пр
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,Ext
               //--- закрашиваем конец свечи (область между последним и предпоследним
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
```

© 2000-2025, MetaQuotes Ltd.

```
               //--- сдвигаем начальную позицию для рисования следующей свечи
               ExtStartPos=i;
               //--- увеличиваем счетчик массива
               ExtCount++;
              }
            else
               continue;
           }
         else
           {
            //--- сбрасываем значения ошибки
            ResetLastError();
            //--- получаем последнюю дату со старшего таймфрейма
            if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
              {
               Print("Ошибка копирования данных, код = ",GetLastError());
               return(0);
              }
            //--- если новая дата больше, то значит заканчиваем формирование свечки
            if(ExtCurrentTime[0]>ExtLastTime)
              {
               //--- очистим область между последним и предпоследним баром в главных и
               ClearEndOfBodyMain(i-1);
               //--- закрасим эту область с помощью вспомогательных индикаторных буфер
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
               //--- сдвигаем начальную позицию для рисования следующей свечи
               ExtStartPos=i;
               //--- сбросим флаг направления цены
               ExtDirectionFlag=0;
               //--- запомним новую последнюю дату
               ExtLastTime=ExtCurrentTime[0];
              }
            else
              {
               //--- формируем свечку
               FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Проверка введенного периода индикатора на корректность           |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- период индикатора должен быть больше таймфрейма, на котором он отображается
```

© 2000-2025, MetaQuotes Ltd.

```
   if(current_period>=high_period)
     {
      Print("Ошибка! Значение периода индикатора должно быть больше значения текущего
      return(false);
     }
//--- если период индикатора - одна неделя или месяц, то период корректен
   if(high_period>32768)
      return(true);
//--- приведем значения периодов к минутам
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- период индикатора должен быть кратен таймфрейму, на котором он отображается
   if(high_period%current_period!=0)
     {
      Print("Ошибка! Значение периода индикатора должно быть кратным значению текущего
      return(false);
     }
//--- период индикатора должен превышать значение таймфрейма, на котором он отображает
   if(high_period/current_period<3)
     {
      Print("Ошибка! Значение периода индикатора должно превышать значение текущего та
      return(false);
     }
//--- период индикатора корректен для текущего таймфрейма
   return(true);
  }
//+------------------------------------------------------------------+
//| Получение данных времени со старшего таймфрейма                  |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- сброс значения ошибки
   ResetLastError();
//--- скопируем все данные на текущее время
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- получим код ошибки
      int code=GetLastError();
      //--- распечатаем текст ошибки
      PrintFormat("Ошибка копирования данных! %s",code==4401
                  ? "История еще подгружается!"
                  : "Код = "+IntegerToString(code));
      //--- вернем false для повторной попытки загрузки данных
      return(false);
     }
//--- получим размер массива
   ExtSize=ArraySize(ExtTimeBuff);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим индекс цикла для массива равным нулю
   ExtCount=0;
//--- установим позицию текущей свечи на данном таймфрейме равной нулю
   ExtStartPos=0;
   ExtStartFlag=true;
//--- запомним последнее значение времени со старшего таймфрейма
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция формирует основную часть свечи. В зависимости от значения|
//| флага, функция определяет, какие данные в какие массивы должны   |
//| записываться для корректного отображения.                         |
//+------------------------------------------------------------------+
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)
  {
//--- проверяем значение флага
   if(flag)
     {
      //--- формируем тело свечи
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- формируем тень свечи
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- формируем тело свечи
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- формируем тень свечи
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------+
//| Функция формирует конец свечи. В зависимости от значения флага,  |
//| функция определяет, какие данные в какие массивы должны          |
//| записываться для корректного отображения.                         |
//+------------------------------------------------------------------+
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)
  {
//--- проверяем значение флага
```

© 2000-2025, MetaQuotes Ltd.

```
   if(flag)
     {
      //--- формируем конец тела свечи
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- формируем конец тени свечи
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- меняем значение флага на противоположное
      flag=false;
     }
   else
     {
      //--- формируем конец тела свечи
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- формируем конец тени свечи
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- меняем значение флага на противоположное
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Очистка конца свечи (область между последним и предпоследним     |
//| баром)                                                           |
//+------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------+
//| Очистка свечи                                                    |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- проверка
   if(count!=0)
     {
      //--- заполняем индикаторные буферы пустым значением
      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);
     }
  }
//+------------------------------------------------------------------+
//| Формирование основной части свечи                                |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- проверка
   if(count!=0)
     {
      //--- заполняем индикаторные буферы значениями
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| Формирование конца свечи                                         |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- заполняем индикаторные буферы значениями
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Заполнение i-ого элемента индикаторных буферов пустыми значениями|
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- устанавливаем пустое значение в ячейку индикаторных буферов
   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;
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayMinimum

Ищет минимальный элемент в первом измерении многомерного числового массива.

```
int  ArrayMinimum(
   const void&   array[],             // массив для поиска
   int           start=0,             // с какого индекса начинаем поиск
   int           count=WHOLE_ARRAY    // количество проверяемых
   );
```

Параметры

array[]

[in] Числовой массив, в котором производится поиск.

start=0

[in] Начальный индекс для поиска.

count=WHOLE_ARRAY

[in] Количество элементов для поиска. По умолчанию, ищет во всем массиве (count=WHOLE_ARRAY).

Возвращаемое значение

Функция возвращает индекс найденного элемента с учетом серийности массива. В случае неудачи функция возвращает -1.

Примечание

Поиск минимального элемента производится с учетом значения флага AS_SERIES.

Функции ArrayMaximum и ArrayMinimum принимают в качестве параметра массив любой размерности, при этом поиск происходит только по первому (нулевому) измерению.

Пример:

```
#property description "Индикатор отображает свечи со старшего таймфрейма на текущем."
//--- настройки индикатора
#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
```

© 2000-2025, 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
//--- предопределенная константа
#define INDICATOR_EMPTY_VALUE 0.0
//--- входные параметры
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Таймфрейм для расчета индик
input datetime        InpDateStart=D'2013.01.01 00:00'; // Дата начала анализа
//--- индикаторные буферы для медвежьих свечей
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- индикаторные буферы для бычьих свечей
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- глобальные переменные
datetime ExtTimeBuff[];      // буфер для времени с высшего таймфрейма
int      ExtSize=0;          // размер буфера времени
int      ExtCount=0;         // индекс внутри буфера времени
int      ExtStartPos=0;      // начальная позиция для расчета индикатора
bool     ExtStartFlag=true;  // вспомогательный флаг для получения начальной позиции
datetime ExtCurrentTime[1];  // последнее время формирования бара со старшего таймфрей
datetime ExtLastTime;        // последнее время со старшего таймфрейма, для которого г
bool     ExtBearFlag=true;   // флаг для определения порядка записи данных в медвежьи
bool     ExtBullFlag=true;   // флаг для определения порядка записи данных в бычьи инд
int      ExtIndexMax=0;      // индекс максимального элемента в массиве
int      ExtIndexMin=0;      // индекс минимального элемента в массиве
int      ExtDirectionFlag=0; // направление движения цены у текущей свечи
//--- отступ между ценой открытия и закрытия свечи для правильной отрисовки
```

© 2000-2025, MetaQuotes Ltd.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//| Закрашивание основной части свечи                                |
//+------------------------------------------------------------------+
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)
  {
//--- найдем индексы максимального и минимального элементов в массивах
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // максимум в High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // минимум в Low
//--- определим сколько баров с текущего таймфрейма будем закрашивать
   int count=fill_index-start+1;
//--- если цена закрытия на первом баре больше цены закрытия на последнем - свеча медв
   if(open[start]>close[last])
     {
      //--- если до этого свеча была бычьей, то очищаем значения бычьих индикаторных б
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- медвежья свеча
      ExtDirectionFlag=-1;
      //--- формируем свечку
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- выход из функции
      return;
     }
//--- если цена закрытия на первом баре меньше цены закрытия на последнем - свеча быч
   if(open[start]<close[last])
     {
      //--- если до этого свеча была медвежьей, то очищаем значения медвежьих индикато
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- бычья свеча
      ExtDirectionFlag=1;
      //--- формируем свечку
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- выход из функции
      return;
     }
//--- если попали в эту часть функции, то значит цена открытия на первом баре равняет
//--- цене закрытия на последнем баре; будем считать такую свечу медвежьей
//--- если до этого свеча была бычьей, то очищаем значения бычьих индикаторных буферо
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- медвежья свеча
   ExtDirectionFlag=-1;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если цены закрытия и цены открытия равны, то используем сдвиг для корректного от
   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);
  }
//+------------------------------------------------------------------+
//| Закрашивание конца свечи                                         |
//+------------------------------------------------------------------+
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)
  {
//--- если всего один бар, то не рисуем
   if(last-start==0)
      return;
//--- если цена закрытия на первом баре больше цены закрытия на последнем - свеча медв
   if(open[start]>close[last])
     {
      //--- формируем конец свечи
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- выход из функции
      return;
     }
//--- если цена закрытия на первом баре меньше цены закрытия на последнем - свеча быч
   if(open[start]<close[last])
     {
      //--- формируем конец свечи
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- выход из функции
      return;
     }
//--- если попали в эту часть функции, то значит цена открытия на первом баре равняетс
//--- цене закрытия на последнем баре; будем считать такую свечу медвежьей
//--- формируем конец свечи
   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
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- проверка периода индикатора
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- отображение ценовых данных на переднем плане
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- привязка индикаторных буферов
   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);
//--- зададим некоторые значения свойств для построения индикатора
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // тип графического построен
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // стиль линии отрисовки
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // толщина линии отрисовки
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- если еще нет рассчитанных баров
   if(prev_calculated==0)
     {
      //--- получим время появления баров со старшего таймфрейма
      if(!GetTimeData())
         return(0);
     }
//--- установим прямое направление индексации
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- переменная начала для расчета баров
   int start=prev_calculated;
//--- если бар формируется, то пересчитываем значение индикатора на нем
   if(start!=0 && start==rates_total)
      start--;
//--- цикл расчета значений индикатора
   for(int i=start;i<rates_total;i++)
     {
      //--- заполняем i-ые элементы индикаторных буферов пустыми значениями
      FillIndicatorBuffers(i);
      //--- проводим вычисление для баров начиная с даты InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- в первый раз определим позицию с которой начнем отображать значения
         if(ExtStartFlag)
           {
            //--- запомним номер начального бара
            ExtStartPos=i;
            //--- определим первую дату со старшего таймфрейма, которая больше time[i]
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- изменим значение флага, чтобы больше не попадать в этот блок
            ExtStartFlag=false;
           }
         //--- проверка, есть ли еще в массиве элементы
         if(ExtCount<ExtSize)
           {
            //--- ждем, когда значение времени с текущего таймфрейма достигнет значени
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- рисуем главную часть свечи (без закрашивания между последним и пр
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,Ext
               //--- закрашиваем конец свечи (область между последним и предпоследним
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
```

© 2000-2025, MetaQuotes Ltd.

```
                //--- сдвигаем начальную позицию для рисования следующей свечи
                ExtStartPos=i;
                //--- увеличиваем счетчик массива
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- сбрасываем значения ошибки
             ResetLastError();
             //--- получаем последнюю дату со старшего таймфрейма
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Ошибка копирования данных, код = ",GetLastError());
                return(0);
               }
             //--- если новая дата больше, то значит заканчиваем формирование свечки
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- очистим область между последним и предпоследним баром в главных и
                ClearEndOfBodyMain(i-1);
                //--- закрасим эту область с помощью вспомогательных индикаторных буфер
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- сдвигаем начальную позицию для рисования следующей свечи
                ExtStartPos=i;
                //--- сбросим флаг направления цены
                ExtDirectionFlag=0;
                //--- запомним новую последнюю дату
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- формируем свечку
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Проверка введенного периода индикатора на корректность           |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- период индикатора должен быть больше таймфрейма, на котором он отображается
```

© 2000-2025, MetaQuotes Ltd.

```
   if(current_period>=high_period)
     {
      Print("Ошибка! Значение периода индикатора должно быть больше значения текущего
      return(false);
     }
//--- если период индикатора - одна неделя или месяц, то период корректен
   if(high_period>32768)
      return(true);
//--- приведем значения периодов к минутам
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- период индикатора должен быть кратен таймфрейму, на котором он отображается
   if(high_period%current_period!=0)
     {
      Print("Ошибка! Значение периода индикатора должно быть кратным значению текущего
      return(false);
     }
//--- период индикатора должен превышать значение таймфрейма, на котором он отображает
   if(high_period/current_period<3)
     {
      Print("Ошибка! Значение периода индикатора должно превышать значение текущего та
      return(false);
     }
//--- период индикатора корректен для текущего таймфрейма
   return(true);
  }
//+------------------------------------------------------------------+
//| Получение данных времени со старшего таймфрейма                  |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- сброс значения ошибки
   ResetLastError();
//--- скопируем все данные на текущее время
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- получим код ошибки
      int code=GetLastError();
      //--- распечатаем текст ошибки
      PrintFormat("Ошибка копирования данных! %s",code==4401
                  ? "История еще подгружается!"
                  : "Код = "+IntegerToString(code));
      //--- вернем false для повторной попытки загрузки данных
      return(false);
     }
//--- получим размер массива
   ExtSize=ArraySize(ExtTimeBuff);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим индекс цикла для массива равным нулю
   ExtCount=0;
//--- установим позицию текущей свечи на данном таймфрейме равной нулю
   ExtStartPos=0;
   ExtStartFlag=true;
//--- запомним последнее значение времени со старшего таймфрейма
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Функция формирует основную часть свечи. В зависимости от значения|
//| флага, функция определяет, какие данные в какие массивы должны   |
//| записываться для корректного отображения.                         |
//+------------------------------------------------------------------+
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)
  {
//--- проверяем значение флага
   if(flag)
     {
      //--- формируем тело свечи
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- формируем тень свечи
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- формируем тело свечи
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- формируем тень свечи
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------+
//| Функция формирует конец свечи. В зависимости от значения флага,  |
//| функция определяет, какие данные в какие массивы должны          |
//| записываться для корректного отображения.                         |
//+------------------------------------------------------------------+
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)
  {
//--- проверяем значение флага
```

© 2000-2025, MetaQuotes Ltd.

```
   if(flag)
     {
      //--- формируем конец тела свечи
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- формируем конец тени свечи
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- меняем значение флага на противоположное
      flag=false;
     }
   else
     {
      //--- формируем конец тела свечи
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- формируем конец тени свечи
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- меняем значение флага на противоположное
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Очистка конца свечи (область между последним и предпоследним     |
//| баром)                                                           |
//+------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------+
//| Очистка свечи                                                    |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- проверка
   if(count!=0)
     {
      //--- заполняем индикаторные буферы пустым значением
      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);
     }
  }
//+------------------------------------------------------------------+
//| Формирование основной части свечи                                |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- проверка
   if(count!=0)
     {
      //--- заполняем индикаторные буферы значениями
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| Формирование конца свечи                                         |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- заполняем индикаторные буферы значениями
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Заполнение i-ого элемента индикаторных буферов пустыми значениями|
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- устанавливаем пустое значение в ячейку индикаторных буферов
   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;
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayPrint

Выводит в журнал массив простого типа или простой структуры.

```
void  ArrayPrint(
   const void&   array[],             // выводимый массив
   uint          digits=_Digits,      // количество десятичных знаков после запятой
   const string  separator=NULL,      // разделитель между значениями полей структуры
   ulong         start=0,             // индекс первого выводимого элемента
   ulong         count=WHOLE_ARRAY,   // количество выводимых элементов
   ulong         flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_
   );
```

Параметры

array[]

[in] Массив простого типа или простой структуры.

digits=_Digits

[in] Количество знаков после запятой для вещественных типов. По умолчанию равно _Digits.

separator=NULL

[in] Разделитель между значениями полей элемента структуры. Значение по умолчанию NULL означает пустую строку, в этом случае разделителем является пробел.

start=0

[in] Индекс первого выводимого элемента массива. По умолчанию выводится с нулевого индекса.

count=WHOLE_ARRAY

[in] Количество элементов массива, которые нужно вывести. По умолчанию выводится весь массив (count=WHOLE_ARRAY).

flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_ALIGN

[in] Комбинация флагов, задающая режим вывода. По умолчанию включены все флаги:

- ARRAYPRINT_HEADER - вывод заголовков для массива структур
- ARRAYPRINT_INDEX - вывод слева номера индекса
- ARRAYPRINT_LIMIT - вывод только 100 первых и 100 последних элементов массива. Используется, если нужно вывести только часть большого массива.
- ARRAYPRINT_ALIGN - включить выравнивание выводимых значений - числа будут выравниваться вправо, строки влево.
- ARRAYPRINT_DATE - при выводе datetime выводить дату в формате dd.mm.yyyy
- ARRAYPRINT_MINUTES - при выводе datetime выводить время в формате HH:MM
- ARRAYPRINT_SECONDS - при выводе datetime выводить время в формате HH:MM:SS

Возвращаемое значение

Нет

Примечание

ArrayPrint() выводит в журнал не все поля массива структур - поля-массивы и поля-указатели объектов пропускаются. Эти столбцы просто не будут выведены на печать для поддержания

© 2000-2025, MetaQuotes Ltd.

простого и удобного представления. Если вам нужен вывод всех полей такой структуры, то вам необходимо написать собственную функцию массового вывода с желаемым форматированием.

Пример:

```
//--- выводит значения 10 последних баров
   MqlRates rates[];
   if(CopyRates(_Symbol,_Period,1,10,rates))
     {
      ArrayPrint(rates);
      Print("Проверка\n[time]\t[open]\t[high]\t[low]\t[close]\t[tick_volume]\t[spread]
      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());
//--- пример вывода
/*
                    [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   17
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.11930 1.12747         17829        9   15
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744         13458       10    9
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194         15362        9   12
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172         16833        9   12
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052         15933        8   10
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528         11888        9    8
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915          7284       10    5
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904          8710        9    6
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263          8956        7    7
   Проверка
   [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
*/
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

FileSave, FileLoad

© 2000-2025, MetaQuotes Ltd.

ArrayRange

Возвращает число элементов в указанном измерении массива.

```
int  ArrayRange(
   const void&   array[],      // массив для проверки
   int           rank_index    // номер измерения
   );
```

Параметры

array[]

[in] Проверяемый массив.

rank_index

[in] Индекс измерения.

Возвращаемое значение

Число элементов в указанном измерении массива

Примечание

Поскольку индексы начинаются с нуля, количество измерений массива на единицу больше, чем индекс последнего измерения.

Пример:

```
void OnStart()
  {
//--- создадим четырехмерный массив
   double array[][5][2][4];
//--- зададим размер нулевого измерения
   ArrayResize(array,10,10);
//--- распечатаем размерности измерений
   int temp;
   for(int i=0;i<4;i++)
     {
      //--- получим размер i-ого измерения
      temp=ArrayRange(array,i);
      //--- распечатаем
      PrintFormat("dim = %d, range = %d",i,temp);
     }
//--- Результат
// dim = 0, range = 10
// dim = 1, range = 5
// dim = 2, range = 2
// dim = 3, range = 4
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayResize

Устанавливает новый размер в первом измерении массива

```
int  ArrayResize(
   void&  array[],              // массив, переданный по ссылке
   int    new_size,             // новый размер массива
   int    reserve_size=0        // резервное значение размера (избыточное)
   );
```

Параметры

array[]

[out] Массив для изменения размеров.

new_size

[in] Новый размер для первого измерения.

reserve_size=0

[in] Размер для дополнительного резерва.

Возвращаемое значение

При успешном выполнении функция возвращает количество всех элементов, содержащихся в массиве после изменения размера; в противном случае возвращает -1 и массив не меняет размеры.

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.

Примечание

Функция может быть применена только к динамическим массивам. При этом необходимо иметь ввиду, что нельзя изменять размер для динамических массивов, назначенных в качестве индикаторных буферов функцией SetIndexBuffer(). Для индикаторных буферов все операции по изменению размера производит исполняющая подсистема терминала.

Общее число элементов в массиве не может превышать 2147483647.

При частом распределении памяти рекомендуется использовать третий параметр, задающий резерв для уменьшения количества физического распределения памяти. Все последующие вызовы функции ArrayResize не приводят к физическому перераспределению памяти, а только меняется размер первого измерения массива в пределах зарезервированной памяти. Следует помнить, что третий параметр будет использоваться только тогда, когда будет происходить физическое распределение памяти, например:

```
ArrayResize(arr,1000,1000);
for(int i=1;i<3000;i++)
   ArrayResize(arr,i,1000);
```

В данном случае произойдёт 2 перераспределения памяти, один раз до входа в цикл на 3000 итераций, при этом размерность массива будет установлена в 1000 и второй при i равной 2000.

© 2000-2025, MetaQuotes Ltd.

Если третий параметр опустить, то произойдёт 2000 физических перераспределения памяти и это замедлит выполнение программы.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- счетчики
   ulong start=GetTickCount();
   ulong now;
   int   count=0;
//--- массив для демонстрации быстрого варианта
   double arr[];
   ArrayResize(arr,100000,100000);
//--- проверим, как быстро работает вариант с резервированием памяти
   Print("--- Test Fast: ArrayResize(arr,100000,100000)");
   for(int i=1;i<=300000;i++)
     {
      //--- задаем новый размер массива с указанием резерва в 100000 элементов!
      ArrayResize(arr,i,100000);
      //--- при достижении круглой цифры выводим размер массива и затраченное время
      if(ArraySize(arr)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-sta
         start=now;
        }
     }
//--- покажем теперь, как медленно работает вариант без резервирования памяти
   double slow[];
   ArrayResize(slow,100000,100000);
//---
   count=0;
   start=GetTickCount();
   Print("---- Test Slow: ArrayResize(slow,100000)");
//---
   for(int i=1;i<=300000;i++)
     {
      //--- задаем новый размер массива, но уже без дополнительного резерва
      ArrayResize(slow,i);
      //--- при достижении круглой цифры выводим размер массива и затраченное время
      if(ArraySize(slow)%100000==0)
        {
         now=GetTickCount();
         count++;
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("%d. ArraySize(slow)=%d Time=%d ms",count,ArraySize(slow),(now-st
         start=now;
        }
     }
  }
//--- Примерный результат выполнения скрипта
/*
   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
*/
```

Смотри также

ArrayInitialize

© 2000-2025, MetaQuotes Ltd.

ArrayInsert

Вставляет в массив-приемник из массива-источника указанное число элементов, начиная с указанного индекса.

```
bool  ArrayInsert(
   void&         dst_array[],          // массив-приемник
   const void&   src_array[],          // массив источник
   uint          dst_start,            // индекс в массиве-приемнике для вставки
   uint          src_start=0,          // индекс в массиве-источнике для копирования
   uint          count=WHOLE_ARRAY     // количество вставляемых элементов
   );
```

Параметры

dst_array[]

[in][out] Массив-приемник, в который необходимо добавить элементы.

src_array[]

[in] Массив-источник, из которого необходимо добавить элементы.

dst_start

[in] Индекс в массиве-приемнике для вставки элементов из массива-источника.

src_start=0

[in] Индекс в массиве-, начиная с которого берутся элементы массива-источника для вставки.

count=WHOLE_ARRAY

[in] Количество добавляемых элементов из массива-источника. Значение WHOLE_ARRAY означает все элементы с указанного индекса до конца массива.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 5052 - ERR_SMALL_ARRAY (параметры *start* и/или *count* заданы неверно или исходный массив *src_array[]* является пустым),
- 5056 - ERR_SERIES_ARRAY (массив не может быть изменен, индикаторный буфер),
- 4006 - ERR_INVALID_ARRAY (копирование в себя недопустимо, либо массивы имеют разный тип, либо массив фиксированного размера, который содержит объекты класса или структуры с деструктором),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (массив содержит не POD-структуры, то есть простое копирование невозможно),
- ошибки изменения размера массива-приемника *dst_array[]* - они приведены в описании функции ArrayRemove().

Примечание

Если функция используется для массива фиксированного размера, то сам размер массива-приемника *dst_array[]* не меняется, при этом начиная с позиции *dst_start* элементы массива-

© 2000-2025, MetaQuotes Ltd.

приемника сдвигаются вправо (последние *count* элементов "выпадают"), а на освободившееся место происходит копирование элементов из массива-источника.

Нельзя вставлять элементы в динамические массивы, назначенные в качестве индикаторных буферов функцией SetIndexBuffer(). Для индикаторных буферов все операции по изменению размера производит исполняющая подсистема терминала.

В массиве-источнике элементы копируются, начиная с индекса *src_start*. Размер массива-источника при этом не изменяется. Добавляемые в массив-приемник элементы не являются ссылками на элементы массива-источника - это означает, что последующие изменения элементов в любом из двух массивов не отражаются на втором.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим массив фиксированного размера и заполним значениями
   int array_dest[10];
   for(int i=0;i<10;i++)
     {
      array_dest[i]=i;
     }
   //--- массив-источник
   int array_source[10];
   for(int i=0;i<10;i++)
     {
      array_source[i]=10+i;
     }
//--- покажем массивы до вставки элементов
   Print("До вызова ArrayInsert()");
   ArrayPrint(array_dest);
   ArrayPrint(array_source);
//--- вставим 3 элемента из массива-источника и покажем новый состав массив-приемника
   ArrayInsert(array_dest,array_source,4,0,3);
   Print("После вызова ArrayInsert()");
   ArrayPrint(array_dest);
/*
   Результат выполнения
   До вызова ArrayInsert()
   0 1 2 3 4 5 6 7 8 9
   После вызова ArrayInsert()
   0 1 2 3 10 11 12 7 8 9
*/
```

Смотри также

ArrayRemove, ArrayCopy, ArrayResize, ArrayFree

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

ArrayRemove

Удаляет из массива указанное число элементов начиная с указанного индекса.

```
bool  ArrayRemove(
   void&         array[],            // массив любого типа
   uint          start,              // с какого индекса начинаем удалять
   uint          count=WHOLE_ARRAY   // количество элементов
   );
```

Параметры

array[]

[in][out] Массив.

start

[in] Индекс, начиная с которого удаляются элементы массива.

count=WHOLE_ARRAY

[in] Количество удаляемых элементов. Значение WHOLE_ARRAY означает удаление всех элементов с указанного индекса до конца массива.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 5052 - ERR_SMALL_ARRAY (значение *start* слишкое большое),
- 5056 - ERR_SERIES_ARRAY (массив не может быть изменен, индикаторный буфер),
- 4003 - ERR_INVALID_PARAMETER (значение *count* слишком большое),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (массив фиксированного размера, который содержит сложные объекты с деструктором),
- 4006 - ERR_INVALID_ARRAY (массив фиксированного размера, который содержит объекты структур или классов с деструктором).

Примечание

Если функция используется для массива фиксированного размера, то сам размер массива не меняется: при этом происходит физическое копирование оставшегося "хвоста" в позицию *start*. Для точного понимания работы функции смотрите пример ниже. "Физическое" копирование означает, что копируемые объекты не создаются с помощью вызова конструктора или оператора копирования, а просто происходит копирование бинарного представления объекта. Именно по этой причине запрещается применять функцию ArrayRemove() к массиву фиксированного размера, содержащего объекты с деструктором (взводится ошибка ERR_INVALID_ARRAY или ERR_STRUCT_WITHOBJECTS_ORCLASS). Так как при удалении такого объекта деструктор должен быть вызван дважды - для первоначального объекта и его копии.

Нельзя удалять элементы из динамических массивов, назначенных в качестве индикаторных буферов функцией SetIndexBuffer(), это приведет к появлению ошибки ERR_SERIES_ARRAY. Для индикаторных буферов все операции по изменению размера производит исполняющая подсистема терминала.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим массив фиксированного размера и заполним значениями
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- покажем массив до удаления элементов
   Print("До вызова ArrayRemove()");
   ArrayPrint(array);
//--- удалим 2 элемента из массива и покажем новый состав
   ArrayRemove(array,4,2);
   Print("После вызова ArrayRemove()");
   ArrayPrint(array);
/*
   Результат выполнения:
   До вызова ArrayRemove()
   0 1 2 3 4 5 6 7 8 9
   После вызова ArrayRemove()
   0 1 2 3 6 7 8 9 8 9
*/
```

Смотри также

ArrayInsert, ArrayCopy, ArrayResize, ArrayFree

© 2000-2025, MetaQuotes Ltd.

ArrayReverse

Разворачивает в массиве указанное число элементов начиная с указанного индекса.

```
bool  ArrayReverse(
   void&         array[],            // массив любого типа
   uint          start=0,            // с какого индекса начинаем переворачивать масси
   uint          count=WHOLE_ARRAY   // количество элементов
   );
```

Параметры

array[]

[in][out] Массив.

start=0

[in] Индекс, начиная с которого разворачивается массив.

count=WHOLE_ARRAY

[in] Количество разворачиваемых элементов. Если указано значение WHOLE_ARRAY, то будут зеркально перемещены между собой все элементы массива, начиная с указанного индекса *start* и до конца массива.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Функция ArraySetAsSeries() физически не перемещает элементы массива, а только меняет направление индексации задом наперед для организации доступа к элементам как в таймсерии. Функция ArrayReverse() физически перемещает элементы массива таким образом, что массив "переворачивается".

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим массив фиксированного размера и заполним значениями
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- покажем массив до разворачивания элементов
   Print("До вызова ArrayReverse()");
   ArrayPrint(array);
//--- развернем 3 элемента в массиве и покажем новый состав
   ArrayReverse(array,4,3);
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("После вызова ArrayReverse()");
   ArrayPrint(array);
/*
   Результат выполнения:
   До вызова ArrayReverse()
   0 1 2 3 4 5 6 7 8 9
   После вызова ArrayReverse()
   0 1 2 3 6 5 4 7 8 9
*/
```

Смотри также

ArrayInsert, ArrayRemove, ArrayCopy, ArrayResize, ArrayFree, ArrayGetAsSeries, ArraySetAsSeries

© 2000-2025, MetaQuotes Ltd.

ArraySetAsSeries

Устанавливает флаг AS_SERIES указанному объекту динамического массива, индексация элементов массива будет производиться как в таймсериях.

```
bool  ArraySetAsSeries(
   const void&  array[],     // массив по ссылке
   bool         flag         // true означает обратный порядок индексации
   );
```

Параметры

array[]

[in][out] Числовой массив для установки.

flag

[in] Направление индексирования массива.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Флаг AS_SERIES не может быть установлен у многомерных массивов и у статических массивов (то есть массивов, чей размер в квадратных скобках указан еще на этапе компиляции). Индексация в таймсерии отличается от обычного массива тем, что индексация элементов таймсерии производится от конца массива к началу (от самых свежих данных к самым старым).

Пример: индикатор, показывающий номер бара




```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Numeration
#property indicator_label1  "Numeration"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_type1   DRAW_LINE
#property indicator_color1  CLR_NONE
//--- indicator buffers
double         NumerationBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- установим индексацию для буфера как в таймсерии
   ArraySetAsSeries(NumerationBuffer,true);
//--- установим точность отображения в DataWindow
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- как будет выглядеть в DataWindow имя индикаторного массива
   PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//---  будем хранить время открытия текущего нулевого бара
   static datetime currentBarTimeOpen=0;
//--- перевернем доступ к массиву time[] - сделаем как в таймсерии
   ArraySetAsSeries(time,true);
//--- если время нулевого бара отличается от того, что мы храним
   if(currentBarTimeOpen!=time[0])
     {
     //--- пронумеруем все бары от текущего момента вглубь графика
      for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
      currentBarTimeOpen=time[0];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

Доступ к таймсериям, ArrayGetAsSeries

© 2000-2025, MetaQuotes Ltd.

ArraySize

Возвращает количество элементов указанного массива.

```
int  ArraySize(
   const void&  array[]    // проверяемый массив
   );
```

Параметры

array[]

[in] Массив любого типа.

Возвращаемое значение

Значение типа int.

Примечание

Для одномерного массива значение, возвращаемое функцией ArraySize, равно значению ArrayRange(array,0).

Пример:

```
void OnStart()
  {
//--- создание массивов
   double one_dim[];
   double four_dim[][10][5][2];
//--- размеры
   int one_dim_size=25;
   int reserve=20;
   int four_dim_size=5;
//--- вспомогательная переменная
   int size;
//--- выделим память без резервирования
   ArrayResize(one_dim,one_dim_size);
   ArrayResize(four_dim,four_dim_size);
//--- 1. одномерный массив
   Print("+============================================================+");
   Print("Размеры массивов:");
   Print("1. Одномерный массив");
   size=ArraySize(one_dim);
   PrintFormat("Размер нулевого измерения = %d, Размер массива = %d",one_dim_size,size
//--- 2. многомерный массив
   Print("2. Многомерный массив");
   size=ArraySize(four_dim);
   PrintFormat("Размер нулевого измерения = %d, Размер массива = %d",four_dim_size,siz
//--- размеры измерений
   int d_1=ArrayRange(four_dim,1);
   int d_2=ArrayRange(four_dim,2);
   int d_3=ArrayRange(four_dim,3);
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("Проверка:");
   Print("Нулевое измерение = Размер массива / (Первое измерение * Второе измерение *
   PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3. одномерный массив с резервированием памяти
   Print("3. Одномерный массив с резервированием памяти");
//--- увеличим значение в 2 раза
   one_dim_size*=2;
//--- выделим память с резервированием
   ArrayResize(one_dim,one_dim_size,reserve);
//--- распечатаем размер
   size=ArraySize(one_dim);
   PrintFormat("Размер с резервированием = %d, Реальный размер массива = %d",one_dim_s
  }
```

© 2000-2025, MetaQuotes Ltd.

ArraySort

Сортирует многомерный числовой массив по возрастанию значений в первом измерении.

```
bool  ArraySort(
   void&  array[]      // массив для сортировки
   );
```

Параметры

array[]

[in][out] Числовой массив для сортировки.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Массив всегда сортируется по возрастанию, независимо от значения флага AS_SERIES.

Функции ArraySort и ArrayBSearch принимают в качестве параметра массив любой размерности, при этом сортировка и поиск происходят только по первому (нулевому) измерению.

Пример:

```
#property description "Индикатор анализирует данные за последний месяц и раскрашивает
#property description "и большими тиковыми объемами. Для определения таких свечей про
#property description "массива тиковых объемов. Свечи, объемы которых составляют перв
#property description "процентов массива, считаются малыми. Свечи, тиковые объемы кот
#property description "последние InpBigVolume процентов массива, считаются большими."
//--- настройки индикатора
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot
#property indicator_label1  "VolumeFactor"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrDodgerBlue,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- предопределенная константа
#define INDICATOR_EMPTY_VALUE 0.0
//--- входные параметры
input int InpSmallVolume=15; // Процент малых объемов (<50)
input int InpBigVolume=80;   // Процент больших объемов (<50)
//--- время начала анализа (будет смещаться)
datetime ExtStartTime;
//--- индикаторные буферы
double   ExtOpenBuff[];
double   ExtHighBuff[];
double   ExtLowBuff[];
```

© 2000-2025, MetaQuotes Ltd.

```
double   ExtCloseBuff[];
double   ExtColorBuff[];
//--- граничные значения объемов для отображения свечей
long     ExtLeftBorder=0;
long     ExtRightBorder=0;
//+------------------------------------------------------------------+
//| Получение значений границ для тиковых объемов                    |
//+------------------------------------------------------------------+
bool GetVolumeBorders(void)
  {
//--- переменные
   datetime stop_time;  // время окончания копирования
   long     buff[];     // буфер, куда будем копировать
//--- время окончания - текущее время
   stop_time=TimeCurrent();
//--- время начала - на месяц раньше от текущего
   ExtStartTime=GetStartTime(stop_time);
//--- получим значения тиковых объемов
   ResetLastError();
   if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
     {
      //--- не удалось получить данные, вернем false для запуска команды на пересчет
      PrintFormat("Не удалось получить значения тикового объема. Код ошибки = %d",GetL
      return(false);
     }
//--- вычислим размер массива
   int size=ArraySize(buff);
//--- отсортируем массив
   ArraySort(buff);
//--- определим значения левой и правой границы для тиковых объемов
   ExtLeftBorder=buff[size*InpSmallVolume/100];
   ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//--- успешное выполнение
   return(true);
  }
//+------------------------------------------------------------------+
//| Получение даты на месяц меньше чем дата во входном параметре     |
//+------------------------------------------------------------------+
datetime GetStartTime(const datetime stop_time)
  {
//--- преобразуем время окончания к переменной структуры типа MqlDateTime
   MqlDateTime temp;
   TimeToStruct(stop_time,temp);
//--- получим дату на месяц меньше
   if(temp.mon>1)
      temp.mon-=1;  // текущий месяц не первый в году, значит номер предыдущего на оди
   else
     {
      temp.mon=12;  // текущий месяц первый в году, значит номер предыдущего равен 12
```

© 2000-2025, MetaQuotes Ltd.

```
      temp.year-=1; // а номер года на один меньше
     }
//--- число дня будет не больше 28
   if(temp.day>28)
      temp.day=28;
//--- вернем полученную дату
   return(StructToTime(temp));
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- проверка, удовлетворяют ли входные параметры условиям
   if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
     {
      Print("Некорректные входные параметры");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,ExtOpenBuff);
   SetIndexBuffer(1,ExtHighBuff);
   SetIndexBuffer(2,ExtLowBuff);
   SetIndexBuffer(3,ExtCloseBuff);
   SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- зададим значение, которое не будет отображаться
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- зададим надписи для индикаторных буферов
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- проверка, есть ли еще необработанные бары
   if(prev_calculated<rates_total)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- получение новых значений левой и правой границы для объемов
      if(!GetVolumeBorders())
         return(0);
     }
//--- переменная начала для расчета баров
   int start=prev_calculated;
//--- если значения индикатора уже были рассчитаны на предыдущем тике, то работаем на
   if(start>0)
      start--;
//--- установим прямое направление индексации в таймсериях
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
   ArraySetAsSeries(tick_volume,false);
//--- цикл расчета значений индикатора
   for(int i=start;i<rates_total;i++)
     {
      //--- закрашиваем свечи, начиная с начальной даты
      if(ExtStartTime<=time[i])
        {
         //--- если значение не меньше правой границы, то закрашиваем свечу
         if(tick_volume[i]>=ExtRightBorder)
           {
            //--- получим данные для рисования свечи
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- цвет DodgerBlue
            ExtColorBuff[i]=0;
            //--- продолжаем цикл
            continue;
           }
         //--- если значение не больше левой границы, то закрашиваем свечу
         if(tick_volume[i]<=ExtLeftBorder)
           {
            //--- получим данные для рисования свечи
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- цвет Orange
            ExtColorBuff[i]=1;
            //--- продолжаем цикл
            continue;
           }
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- для баров, не попавших в расчет, ставим пустое значение
      ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

ArrayBsearch

© 2000-2025, MetaQuotes Ltd.

ArraySwap

Обменивает между собой содержимое двух динамических массивов одного типа. Для многомерных массивов количество элементов во всех измерениях кроме первого должно совпадать.

```
bool  ArraySwap(
   void&  array1[],      // первый массив
   void&  array2[]       // второй массив
   );
```

Параметры

array1[]

[in][out] Массив числового типа.

array2[]

[in][out] Массив числового типа.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. В этом случае GetLastError() вернет код ошибки ERR_INVALID_ARRAY.

Примечание

Функция принимает динамические массивы одинакового типа и одинаковых размерностей, кроме первой. Для целых типов знак игнорируется, т.е. char==uchar)

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- массивы для хранения котировок
   double source_array[][8];
   double   dest_array[][8];
   MqlRates rates[];
//--- получаем данные 20 последних свечей на текущем таймфрейме
   int copied=CopyRates(NULL,0,0,20,rates);
   if(copied<=0)
     {
      PrintFormat("CopyRates(%s,0,0,20,rates) failed, error=%d",
                  Symbol(),GetLastError());
      return;
     }
//--- зададим размер массива под количество скопированных данных
   ArrayResize(source_array,copied);
//--- заполним массив rate_array_1[] данными из rates[]
   for(int i=0;i<copied;i++)
```

© 2000-2025, 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;
      }
//--- произведем обмен данными между source_array[] и dest_array[]
   if(!ArraySwap(source_array,dest_array))
     {
      PrintFormat("ArraySwap(source_array,rate_array_2) failed, error code=%d",GetLast
      return;
     }
//--- убедимся, что после обмена размер массива-источника стал нулевым
   PrintFormat("ArraySwap() done: ArraySize(source_array)=%d",ArraySize(source_array))
//--- выведем данные массива-приемника dest_array[]
   ArrayPrint(dest_array);
  }
```

Смотри также

ArrayCopy, ArrayFill, ArrayRange, ArrayIsDynamic

© 2000-2025, MetaQuotes Ltd.

ArrayToFP16

Производит копирование массива типа float или double в массив типа ushort с заданным форматом.

```
bool   ArrayToFP16(
   const ushort&         dst_array[],        // куда копируем
   const float&          src_array[],        // откуда копируем
   ENUM_FLOAT16_FORMAT   fmt                 // формат
   );
```

Перегрузка для типа double

```
bool   ArrayToFP16(
   const ushort&         dst_array[],        // куда копируем
   const double&         src_array[],        // откуда копируем
   ENUM_FLOAT16_FORMAT   fmt                 // формат
   );
```

Параметры

dst_array[]

[out] Массив-приемник типа ushort.

src_array[]

[in] Массив-источник типа float или double.

fmt

[in] Формат копирования из перечисления ENUM_FLOAT16_FORMAT.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Форматы FLOAT16 и BFLOAT16 определены в перечислении ENUM_FLOAT16_FORMAT и используются в MQL5 исключительно только для работы с ONNX моделями.

Функция используется для преобразования входных параметров типа float или doulbe в тип FLOAT16 и BFLOAT16. Далее эти входные параметры используются в функции OnnxRun.

FLOAT16, также известный как половинная точность или формат "half-precision float", использует 16 бит для представления числа с плавающей точкой. Этот формат обеспечивает баланс между точностью и эффективностью вычислений. FLOAT16 широко применяется в глубоком обучении и нейронных сетях, где требуется высокая производительность при обработке больших объемов данных. Этот формат позволяет ускорить вычисления за счет сокращения размера чисел, что особенно важно при обучении глубоких нейронных сетей на графических процессорах (GPU).

© 2000-2025, MetaQuotes Ltd.

BFLOAT16 (или Brain Floating Point 16) также использует 16 бит, но он различается от FLOAT16 в способе представления чисел. В этом формате 8 бит выделены для представления экспоненты, а оставшиеся 7 бит используются для представления мантиссы. Этот формат был разработан для использования в глубоком обучении и искусственном интеллекте, особенно в процессорах Google Tensor Processing Unit (TPU). BFLOAT16 обладает хорошей производительностью при обучении нейронных сетей и может быть эффективно использован для ускорения вычислений.

Пример: функция из статьи Работа с ONNX-моделями в форматах float16 и float8

```
//+------------------------------------------------------------------+
//| RunCastFloat16ToDouble                                           |
//+------------------------------------------------------------------+
bool RunCastFloat16ToDouble(long model_handle)
  {
   PrintFormat("test=%s",__FUNCTION__);
   double test_data[12]= {1,2,3,4,5,6,7,8,9,10,11,12};
   ushort data_uint16[12];
   if(!ArrayToFP16(data_uint16,test_data,FLOAT_FP16))
     {
      Print("error in ArrayToFP16. error code=",GetLastError());
      return(false);
     }
   Print("test array:");
   ArrayPrint(test_data);
   Print("ArrayToFP16:");
   ArrayPrint(data_uint16);
   U<ushort> input_float16_values[3*4];
   U<double> output_double_values[3*4];
   float test_data_float[];
   if(!ArrayFromFP16(test_data_float,data_uint16,FLOAT_FP16))
     {
      Print("error in ArrayFromFP16. error code=",GetLastError());
      return(false);
     }
   for(int i=0; i<12; i++)
     {
      input_float16_values[i].value=data_uint16[i];
      PrintFormat("%d input value =%f  Hex float16 = %s  ushort value=%d",i,test_data_
     }
   Print("ONNX input array:");
   ArrayPrint(input_float16_values);
   bool res=OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float16_values,output_double
   if(!res)
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      return(false);
     }
   Print("ONNX output array:");
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayPrint(output_double_values);
//---
   double sum_error=0.0;
   for(int i=0; i<12; i++)
     {
      double delta=test_data[i]-output_double_values[i].value;
      sum_error+=MathAbs(delta);
      PrintFormat("%d output double %f = %s  difference=%f",i,output_double_values[i].
     }
//---
   PrintFormat("test=%s   sum_error=%f",__FUNCTION__,sum_error);
//---
   return(true);
  }
```

Смотри также

ArrayFromFP16, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

ArrayToFP8

Производит копирование массива типа float или double в массив типа uchar с заданным форматом.

```
bool   ArrayToFP8(
   const uchar&          dst_array[],        // куда копируем
   const float&          src_array[],        // откуда копируем
   ENUM_FLOAT8_FORMAT    fmt                 // формат
   );
```

Перегрузка для типа double

```
bool   ArrayToFP8(
   const uchar&          dst_array[],        // куда копируем
   const double&         src_array[],        // откуда копируем
   ENUM_FLOAT8_FORMAT    fmt                 // формат
   );
```

Параметры

dst_array[]

[out] Массив-приемник типа uchar.

src_array[]

[in] Массив-источник типа float или double.

fmt

[in] Формат копирования из перечисления ENUM_FLOAT8_FORMAT.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Все виды формата FP8 определены в перечислении ENUM_FLOAT8_FORMAT и используются в MQL5 исключительно только для работы с ONNX моделями.

Функция используется для преобразования входных параметров типа float или double в один из типов FP8. Далее эти входные параметры используются в функции OnnxRun.

Формат FP8 (8-битное число с плавающей запятой) представляет собой один из типов данных, используемых для представления чисел с плавающей точкой. В FP8 каждое число представлено 8 битами данных, которые обычно разделяются на три компонента: знак, экспоненту и мантиссу. Этот формат обеспечивает компромисс между точностью и эффективностью хранения данных, что делает его привлекательным для использования в приложениях, где требуется экономия памяти и вычислительных ресурсов.

Благодаря компактному представлению чисел, FP8 позволяет уменьшить требования к памяти и ускорить вычисления. Кроме того, FP8 может быть полезен для реализации низкоуровневых операций, таких как арифметические вычисления и обработка сигналов.

© 2000-2025, MetaQuotes Ltd.

Пример: функция из статьи Работа с ONNX-моделями в форматах float16 и float8

```
//+------------------------------------------------------------------+
//| RunCastFloat8Float                                               |
//+------------------------------------------------------------------+
bool RunCastFloat8ToFloat(long model_handle,const ENUM_FLOAT8_FORMAT fmt)
  {
   PrintFormat("TEST: %s(%s)",__FUNCTION__,EnumToString(fmt));
//---
   float test_data[15]   = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
   uchar data_float8[15] = {};
   if(!ArrayToFP8(data_float8,test_data,fmt))
     {
      Print("error in ArrayToFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   U<uchar> input_float8_values[3*5];
   U<float> output_float_values[3*5];
   float    test_data_float[];
//--- convert float8 to float
   if(!ArrayFromFP8(test_data_float,data_float8,fmt))
     {
      Print("error in ArrayFromFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   for(uint i=0; i<data_float8.Size(); i++)
     {
      input_float8_values[i].value=data_float8[i];
      PrintFormat("%d input value =%f  Hex float8 = %s  ushort value=%d",i,test_data_f
     }
   Print("ONNX input array: ",ArrayToString(input_float8_values));
//--- execute model (convert float8 to float using ONNX)
   if(!OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float8_values,output_float_values
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   Print("ONNX output array: ",ArrayToString(output_float_values));
//--- calculate error (compare ONNX and ArrayFromFP8 results)
   double sum_error=0.0;
   for(uint i=0; i<test_data.Size(); i++)
     {
      double delta=test_data_float[i]-(double)output_float_values[i].value;
      sum_error+=MathAbs(delta);
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("%d output float %f = %s difference=%f",i,output_float_values[i].val
     }
//---
   PrintFormat("%s(%s): sum_error=%f\n",__FUNCTION__,EnumToString(fmt),sum_error);
   return(true);
  }
```

Смотри также

ArrayFromFP8, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

ArrayFromFP16

Производит копирование массива типа ushort в массив типа float или double с заданным форматом.

```
bool   ArrayFromFP16(
   const float&          dst_array[],        // куда копируем
   const ushort&         src_array[],        // откуда копируем
   ENUM_FLOAT16_FORMAT   fmt                 // формат
   );
```

Перегрузка для типа double

```
bool   ArrayFromFP16(
   const double&         dst_array[],        // куда копируем
   const ushort&         src_array[],        // откуда копируем
   ENUM_FLOAT16_FORMAT   fmt                 // формат
   );
```

Параметры

dst_array[]

[out] Массив-приемник типа float или double.

src_array[]

[in] Массив-источник типа ushort.

fmt

[in] Формат копирования из перечисления ENUM_FLOAT16_FORMAT.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Форматы FLOAT16 и BFLOAT16 определены в перечислении ENUM_FLOAT16_FORMAT и используются в MQL5 исключительно только для работы с ONNX моделями.

Если выходные параметры, полученные при выполнении функции OnnxRun, являются типом FLOAT16 и BFLOAT16, то данная функция позволяет конвертировать результат в массивы типа float или double.

FLOAT16, также известный как половинная точность или формат "half-precision float", использует 16 бит для представления числа с плавающей точкой. Этот формат обеспечивает баланс между точностью и эффективностью вычислений. FLOAT16 широко применяется в глубоком обучении и нейронных сетях, где требуется высокая производительность при обработке больших объемов данных. Этот формат позволяет ускорить вычисления за счет сокращения размера чисел, что особенно важно при обучении глубоких нейронных сетей на графических процессорах (GPU).

© 2000-2025, MetaQuotes Ltd.

BFLOAT16 (или Brain Floating Point 16) также использует 16 бит, но он различается от FLOAT16 в способе представления чисел. В этом формате 8 бит выделены для представления экспоненты, а оставшиеся 7 бит используются для представления мантиссы. Этот формат был разработан для использования в глубоком обучении и искусственном интеллекте, особенно в процессорах Google Tensor Processing Unit (TPU). BFLOAT16 обладает хорошей производительностью при обучении нейронных сетей и может быть эффективно использован для ускорения вычислений.

Пример: функция из статьи Работа с ONNX-моделями в форматах float16 и float8

```
//+------------------------------------------------------------------+
//| RunCastFloat16ToDouble                                           |
//+------------------------------------------------------------------+
bool RunCastFloat16ToDouble(long model_handle)
  {
   PrintFormat("test=%s",__FUNCTION__);
   double test_data[12]= {1,2,3,4,5,6,7,8,9,10,11,12};
   ushort data_uint16[12];
   if(!ArrayToFP16(data_uint16,test_data,FLOAT_FP16))
     {
      Print("error in ArrayToFP16. error code=",GetLastError());
      return(false);
     }
   Print("test array:");
   ArrayPrint(test_data);
   Print("ArrayToFP16:");
   ArrayPrint(data_uint16);
   U<ushort> input_float16_values[3*4];
   U<double> output_double_values[3*4];
   float test_data_float[];
   if(!ArrayFromFP16(test_data_float,data_uint16,FLOAT_FP16))
     {
      Print("error in ArrayFromFP16. error code=",GetLastError());
      return(false);
     }
   for(int i=0; i<12; i++)
     {
      input_float16_values[i].value=data_uint16[i];
      PrintFormat("%d input value =%f  Hex float16 = %s  ushort value=%d",i,test_data_
     }
   Print("ONNX input array:");
   ArrayPrint(input_float16_values);
   bool res=OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float16_values,output_double
   if(!res)
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      return(false);
     }
   Print("ONNX output array:");
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayPrint(output_double_values);
//---
   double sum_error=0.0;
   for(int i=0; i<12; i++)
     {
      double delta=test_data[i]-output_double_values[i].value;
      sum_error+=MathAbs(delta);
      PrintFormat("%d output double %f = %s  difference=%f",i,output_double_values[i].
     }
//---
   PrintFormat("test=%s   sum_error=%f",__FUNCTION__,sum_error);
//---
   return(true);
  }
```

Смотри также

ArrayToFP16, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

ArrayFromFP8

Производит копирование массива типа uchar в массив типа float или double с заданным форматом.

```
bool   ArrayFromFP8(
   const float&          dst_array[],        // куда копируем
   const uchar&          src_array[],        // откуда копируем
   ENUM_FLOAT8_FORMAT    fmt                 // формат
   );
```

Перегрузка для типа double

```
bool   ArrayFromFP8(
   const double&         dst_array[],        // куда копируем
   const uchar&          src_array[],        // откуда копируем
   ENUM_FLOAT8_FORMAT    fmt                 // формат
   );
```

Параметры

dst_array[]

[out] Массив-приемник типа float или double.

src_array[]

[in] Массив-источник типа uchar.

fmt

[in] Формат копирования из перечисления ENUM_FLOAT8_FORMAT.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Все виды формата FP8 определены в перечислении ENUM_FLOAT8_FORMAT и используются в MQL5 исключительно только для работы с ONNX моделями.

Если выходные параметры, полученные при выполнении функции OnnxRun, являются типом FP8 из перечисления ENUM_FLOAT8_FORMAT, то данная функция позволяет конвертировать результат в массивы типа float или double.

Формат FP8 (8-битное число с плавающей запятой) представляет собой один из типов данных, используемых для представления чисел с плавающей точкой. В FP8 каждое число представлено 8 битами данных, которые обычно разделяются на три компонента: знак, экспоненту и мантиссу. Этот формат обеспечивает компромисс между точностью и эффективностью хранения данных, что делает его привлекательным для использования в приложениях, где требуется экономия памяти и вычислительных ресурсов.

© 2000-2025, MetaQuotes Ltd.

Благодаря компактному представлению чисел, FP8 позволяет уменьшить требования к памяти и ускорить вычисления. Кроме того, FP8 может быть полезен для реализации низкоуровневых операций, таких как арифметические вычисления и обработка сигналов.

Пример: функция из статьи Работа с ONNX-моделями в форматах float16 и float8

```
//+------------------------------------------------------------------+
//| RunCastFloat8Float                                               |
//+------------------------------------------------------------------+
bool RunCastFloat8ToFloat(long model_handle,const ENUM_FLOAT8_FORMAT fmt)
  {
   PrintFormat("TEST: %s(%s)",__FUNCTION__,EnumToString(fmt));
//---
   float test_data[15]   = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
   uchar data_float8[15] = {};
   if(!ArrayToFP8(data_float8,test_data,fmt))
     {
      Print("error in ArrayToFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   U<uchar> input_float8_values[3*5];
   U<float> output_float_values[3*5];
   float    test_data_float[];
//--- convert float8 to float
   if(!ArrayFromFP8(test_data_float,data_float8,fmt))
     {
      Print("error in ArrayFromFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   for(uint i=0; i<data_float8.Size(); i++)
     {
      input_float8_values[i].value=data_float8[i];
      PrintFormat("%d input value =%f  Hex float8 = %s  ushort value=%d",i,test_data_f
     }
   Print("ONNX input array: ",ArrayToString(input_float8_values));
//--- execute model (convert float8 to float using ONNX)
   if(!OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float8_values,output_float_values
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   Print("ONNX output array: ",ArrayToString(output_float_values));
//--- calculate error (compare ONNX and ArrayFromFP8 results)
   double sum_error=0.0;
   for(uint i=0; i<test_data.Size(); i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       double delta=test_data_float[i]-(double)output_float_values[i].value;
       sum_error+=MathAbs(delta);
       PrintFormat("%d output float %f = %s difference=%f",i,output_float_values[i].val
      }
//---
   PrintFormat("%s(%s): sum_error=%f\n",__FUNCTION__,EnumToString(fmt),sum_error);
   return(true);
  }
```

Смотри также

ArrayToFP8, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

Матрицы и векторы

Матрица — двумерный массив чисел типа double, float или complex.

Вектор — одномерный массив чисел типа double, float или complex. При этом у вектора нет признака, какой он — вертикальный или горизонтальный. Определяется из контекста использования, например, векторная операция Dot предполагает, что левый вектор — горизонтальный, правый вектор — вертикальный. Если нужна совершенная определённость, то можно использовать матрицы с одной строкой или одной колонкой, но как правило, в этом нет необходимости.

У матриц и векторов память под данные распределяется динамически. Фактически, матрицы и векторы являются объектами, которые имеют определённые свойства, такие как тип содержащихся данных и размеры. Свойства матриц и векторов могут быть получены при помощи методов, например, vector_a.Size(), matrix_b.Rows(), vector_c.Norm(), matrix_d.Cond() и т.д. При этом любая размерность может быть изменена.

При создании и инициализации матриц используются так называемые статические методы (это как статические методы класса), например: matrix::Eye(), matrix::Identity(), matrix::Ones(), vector::Ones(), matrix::Zeros(), vector::Zeros(), matrix::Full(), vector::Full(), matrix::Tri().

Для быстрого знакомства с данным разделом читайте статьи:

- Матрицы и векторы в MQL5
- Работа с матрицами и векторами в MQL5

На данный момент операции над матрицами и векторами не предполагают использования типа данных комплекса — в этом направлении работа ещё не закончена.

В MQL5 реализована поддержка передачи матриц и векторов в DLL. Это позволяет импортировать функции, использующие этот тип данных, из внешних библиотек.

Матрицы и векторы передаются в DLL в виде указателя на буфер. Например, для передачи матрицы типа float, соответствующий параметр экспортируемой из DLL-функции должен принимать указатель на буфер с типом 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
```

Для правильной обработки матриц и векторов, кроме их буферов, следует передавать их размеры.

Ниже перечислены все методы матриц и векторов в алфавитном порядке.

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Категория
Activation	Вычисляет значения функции активации и записывает в переданный вектор/матрицу	Машинное обучение
ArgMax	Возвращает индекс максимального значения	Статистика
ArgMin	Возвращает индекс минимального значения	Статистика
ArgSort	Возвращает отсортированный индекс	Манипуляции
Assign	Копирует матрицу, вектор или массив с автоматическим преобразованием	Инициализация
Average	Вычисляет средневзвешенное значение значений матрицы/вектора	Статистика
Cholesky	Вычисляет разложение Холецкого	Преобразования
Clip	Ограничивает элементы матрицы/вектора заданным диапазоном допустимых значений	Манипуляции
Col	Возвращает вектор-столбец. Записывает вектор в указанный столбец	Манипуляции
Cols	Возвращает количество столбцов в матрице	Характеристики
Compare	Сравнивает элементы двух матриц/векторов с заданной точностью	Манипуляции
CompareByDigits	Сравнивает элементы двух матриц/векторов на совпадение с точностью значащих цифр	Манипуляции
Cond	Вычисляет условное число матрицы	Характеристики
Convolve	Возвращает дискретную линейную свертку двух векторов	Произведения
Copy	Возвращает копию заданной матрицы/вектора	Манипуляции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Категория
Concat	Объединяет 2 подматрицы в одну матрицу. Объединяет 2 вектора в один	Манипуляции
CopyIndicatorBuffer	Получает в вектор данные указанного буфера указанного индикатора в указанном количестве	Инициализация
CopyRates	Получает в матрицу или вектор исторические серии структуры MqlRates указанного символа-периода в указанном количестве	Инициализация
CopyTicks	Получает в матрицу или вектор тики из структуры MqlTick	Инициализация
CopyTicksRange	Получает в матрицу или вектор тики из структуры MqlTick в указанном диапазоне дат	Инициализация
CorrCoef	Вычисляет коэффициент корреляции Пирсона (линейный коэффициент корреляции)	Произведения
Correlate	Вычисляет кросс-корреляцию двух векторов	Произведения
Cov	Вычисляет ковариационную матрицу	Произведения
CumProd	Возвращает кумулятивное произведение элементов матрицы/вектора, включая элементы вдоль заданной оси	Статистика
CumSum	Возвращает кумулятивную сумму элементов матрицы/вектора, включая элементы вдоль заданной оси	Статистика
Derivative	Вычисляет значения производной активационной функции и записывает в переданный вектор/матрицу	Машинное обучение
Det	Вычисляет детерминант квадратной невырожденной матрицы	Характеристики
Diag	Извлекает диагональ или строит диагональную матрицу	Манипуляции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Категория
Dot	Скалярное произведение двух векторов	Произведения
Eig	Вычисляет собственные значения и правые собственные векторы квадратной матрицы	Преобразования
EigVals	Вычисляет собственные значения общей матрицы	Преобразования
Eye	Возвращает матрицу с единицами на диагонали и нулями в других местах	Инициализация
Fill	Заполняет существующую матрицу или вектор заданным значением	Инициализация
Flat	Позволяет обращаться к элементу матрицы через один индекс вместо двух	Манипуляции
Full	Создает и возвращает новую матрицу, заполненную заданным значением	Инициализация
GeMM	Общее матричное произведение двух матриц (General Matrix Multiply)	Произведения
HasNan	Возвращает количество NaN значений в матрице/векторе	Манипуляции
Hsplit	Горизонтальное разделение матрицы на несколько подматриц. То же самое, что Split с axis=0	Манипуляции
Identity	Создает единичную матрицу указанного размера	Инициализация
Init	Инициализация матрицы или вектора	Инициализация
Inner	Внутреннее произведение двух матриц	Произведения
Inv	Вычисляет (мультипликативную) обратную квадратной невырожденной матрицы методом Жордана-Гаусса	Решения

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Категория
Kron	Возвращает произведение Кронекера двух матриц, матрицы и вектора, вектора и матрицы или двух векторов	Произведения
LinearRegression	Вычисляет вектор/матрицу с рассчитанными значениями линейной регрессии	Статистика
Loss	Вычисляет значения функции потерь и записывает в переданный вектор/матрицу	Машинное обучение
LstSq	Возвращает решение линейных алгебраических уравнений по методу наименьших квадратов (для неквадратных или вырожденных матриц)	Решения
LU	LU-факторизация матрицы как произведения нижней треугольной матрицы и верхней треугольной матрицы	Преобразования
LUP	Факторизация LUP с частичной перестановкой, которая относится к разложению LU только с перестановкой строк: PA=LU	Преобразования
MatMul	Матричное произведение двух матриц	Произведения
Max	Возвращает максимальное значение в матрице/векторе	Статистика
Mean	Вычисляет среднее арифметическое значений элементов	Статистика
Median	Вычисляет медиану элементов матрицы/вектора	Статистика
Min	Возвращает минимальное значение в матрице/векторе	Статистика
Norm	Возвращает норму матрицы или вектора	Характеристики
Ones	Создает и возвращает новую матрицу, заполненную единицами	Инициализация

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Категория
Outer	Вычисляет внешнее произведение двух матриц или двух векторов	Произведения
Percentile	Возвращает указанный процентиль значений элементов матрицы/вектора или элементов вдоль указанной оси	Статистика
PInv	Вычисляет псевдообратную матрицу методом Мура-Пенроуза	Решения
Power	Возводит квадратную матрицу в целочисленную степень	Произведения
Prod	Возвращает произведение элементов матрицы/вектора, которое также может быть выполнено для заданной оси	Статистика
Ptp	Возвращает диапазон значений матрицы/вектора или заданной оси матрицы	Статистика
QR	Вычисляет qr-факторизацию матрицы	Преобразования
Quantile	Возвращает указанный квантиль значений элементов матрицы/вектора или элементов вдоль указанной оси	Статистика
Random	Статическая функция. Создает и возвращает новую матрицу или вектор, заполненный случайными значениями. Случайные значения генерируются равномерно в заданном диапазоне.	Инициализация
Rank	Возвращает ранг матрицы с помощью метода Гаусса	Характеристики
RegressionMetric	Вычисляет регрессионную метрику как ошибку отклонения от линии регрессии, построенной на указанном массиве данных	Статистика
Reshape	Изменение формы матрицы без изменения ее данных	Манипуляции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Категория
Resize	Возвращает новую матрицу с измененной формой и размером	Манипуляции
Row	Возвращает вектор-строку. Записывает вектор в указанную строку	Манипуляции
Rows	Возвращает количество строк в матрице	Характеристики
Set	Устанавливает значение для элемента вектора по указанному индексу	Манипуляции
Size	Возвращает размер вектора	Характеристики
SLogDet	Вычисляет знак и логарифм определителя матрицы	Характеристики
Solve	Решает линейное матричное уравнение или систему линейных алгебраических уравнений	Решения
Sort	Сортировка по месту	Манипуляции
Spectrum	Вычисляет спектр матрицы как набор ее собственных значений из произведения AT*A	Характеристики
Split	Разделение матрицы на несколько подматриц	Манипуляции
Std	Возвращает стандартное отклонение значений элементов матрицы/вектора или элементов вдоль заданной оси	Статистика
Sum	Возвращает сумму элементов матрицы/вектора, которая также может быть выполнена для заданной оси (осей)	Статистика
SVD	Сингулярное разложение значений	Преобразования
SwapCols	Меняет местами столбцы в матрице	Манипуляции
SwapRows	Меняет местами строки в матрице	Манипуляции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Категория
Trace	Возвращает сумму по диагоналям матрицы	Характеристики
Transpose	Транспонирует (меняет оси местами) и возвращает измененную матрицу	Манипуляции
Tri	Строит матрицу с единицами на заданной и ниже диагоналях и нулями в других местах	Инициализация
TriL	Возвращает копию матрицы с обнуленными элементами над k-й диагональю. Нижнетреугольная матрица	Манипуляции
TriU	Возвращает копию матрицы с обнуленными элементами ниже k-й диагонали. Верхнетреугольная матрица	Манипуляции
Var	Вычисляет дисперсию значений элементов матрицы/вектора	Статистика
Vsplit	Вертикальное разделение матрицы на несколько подматриц. То же самое, что Split с axis=1	Манипуляции
Zeros	Создает и возвращает новую матрицу, заполненную нулями	Инициализация

© 2000-2025, MetaQuotes Ltd.

Типы матриц и векторов

matrix и vector — это специальные типы данных в MQL5, предназначенные для выполнения операций из линейной алгебры. Существуют следующие типы данных:

- matrix — матрица, содержащая элементы типа double
- matrixf — матрица, содержащая элементы типа float
- matrixc — матрица, содержащая элементы типа complex.
- vector — вектор, содержащий элементы типа double
- vectorf — вектор, содержащий элементы типа float
- vectorc — вектор, содержащий элементы типа complex

Для применения в шаблонных функциях можно использовать запись matrix<double>, matrix<float>, matrix<complex>, vector<double>, vector<float>, vector<complex> вместо соответствующих типов.

В MQL5 реализована поддержка передачи матриц и векторов в DLL. Это позволяет импортировать функции, использующие этот тип данных, из внешних библиотек.

Матрицы и векторы передаются в DLL в виде указателя на буфер. Например, для передачи матрицы типа float, соответствующий параметр экспортируемой из DLL-функции должен принимать указатель на буфер с типом 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
```

Для правильной обработки матриц и векторов, кроме их буферов, следует передавать их размеры.

Методы инициализации матриц и векторов

Функция	Действие
Assign	Копирует матрицу, вектор или массив с автоматическим преобразованием
CopyRates	Получает в матрицу или вектор исторические серии структуры MqlRates указанного символа-периода в указанном количестве
Eye	Возвращает матрицу с единицами по диагонали и нулями в остальных местах
Identity	Создает единичную матрицу указанного размера
Ones	Создает и возвращает новую матрицу, заполненную единицами
Zeros	Создает и возвращает новую матрицу, заполненную нулями

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
Full	Создает и возвращает новую матрицу, заполненную заданными значениями
Tri	Создает матрицу с единицами по указанной диагонали и ниже нее и нулями в остальных местах
Init	Инициализирует матрицу или вектор
Fill	Заполняет существующую матрицу или вектор заданным значением

© 2000-2025, MetaQuotes Ltd.

Перечисления для работы с матрицами и векторами

В этом разделе описаны перечисления, которые используются в различных методах работы с матрицами и векторами.

ENUM_AVERAGE_MODE

Перечисление типов усреднения.

Идентификатор	Описание
AVERAGE_NONE	Никакого усреднения. Результаты отдаются по каждой метке отдельно
AVERAGE_BINARY	Результат по метке 1 для двоичной классификации
AVERAGE_MICRO	Результат по усреднённой матрице ошибок (confusion matrix)
AVERAGE_MACRO	Усреднённый результат из результатов по матрицам ошибок каждой метки
AVERAGE_WEIGHTED	Взвешенный усреднённый результат

ENUM_VECTOR_NORM

Перечисление векторных норм для vector::Norm.

Идентификатор	Описание
VECTOR_NORM_INF	Бесконечная норма
VECTOR_NORM_MINUS_INF	Отрицательная бесконечная норма
VECTOR_NORM_P	P-норма

ENUM_MATRIX_NORM

Перечисление матричных норм для matrix::Norm и для получения числа обусловленности матрицы matrix::Cond.

Идентификатор	Описание
MATRIX_NORM_FROBENIUS	Норма Фробениуса
MATRIX_NORM_SPECTRAL	Спектральная норма
MATRIX_NORM_NUCLEAR	Ядерная норма
MATRIX_NORM_INF	Бесконечная норма

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
MATRIX_NORM_P1	Норма P1
MATRIX_NORM_P2	Норма P2
MATRIX_NORM_MINUS_INF	Отрицательная бесконечная норма
MATRIX_NORM_MINUS_P1	Отрицательная норма P1
MATRIX_NORM_MINUS_P2	Отрицательная норма P2

ENUM_VECTOR_CONVOLVE

Перечисление для конволюции vector::Convolve и кросс-корреляции vector::Correlate.

Идентификатор	Описание
VECTOR_CONVOLVE_FULL	Полная свертка
VECTOR_CONVOLVE_SAME	Конволюция с типом same
VECTOR_CONVOLVE_VALID	Конволюция с типом valid

ENUM_REGRESSION_METRIC

Перечисление регрессионных метрик для vector::RegressionMetric.

Идентификатор	Описание
REGRESSION_MAE	Средняя абсолютная ошибка
REGRESSION_MSE	Среднеквадратичная ошибка
REGRESSION_RMSE	Корень из среднеквадратичной ошибки
REGRESSION_R2	R-квадрат
REGRESSION_MAPE	Средняя абсолютная процентная ошибка
REGRESSION_MSPE	Среднеквадратичная процентная ошибка
REGRESSION_RMSLE	Корень из среднеквадратичной логарифмической ошибки
REGRESSION_SMAPE	Симметричная средняя абсолютная процентная ошибка
REGRESSION_MAXE	Максимальная абсолютная ошибка
REGRESSION_MEDE	Медианная абсолютная ошибка
REGRESSION_MPD	Среднее пуассоновское отклонение
REGRESSION_MGD	Среднее гамма-отклонение

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
REGRESSION_EXPV	Объяснённая дисперсия

ENUM_CLASSIFICATION_METRIC

Перечисление метрик для задач классификации.

Идентификатор	Описание
CLASSIFICATION_ACCURACY	Общая точность предсказаний по всем классам, доля правильных ответов
CLASSIFICATION_AVERAGE_PRECISION	Усреднённая точность модели
CLASSIFICATION_BALANCED_ACCURACY	Сбалансированная точность предсказаний
CLASSIFICATION_F1	F1-мера. Гармоническое среднее между точностью (precision) и полнотой(recall) модели
CLASSIFICATION_JACCARD	Мера Жаккара
CLASSIFICATION_PRECISION	Точность модели
CLASSIFICATION_RECALL	Полнота модели
CLASSIFICATION_ROC_AUC	Площадь под кривой ошибок
CLASSIFICATION_TOP_K_ACCURACY	Частота появления правильной метки в топе из k предсказанных меток

ENUM_LOSS_FUNCTION

Перечисление для расчёта функции потерь vector::Loss.

Идентификатор	Описание
LOSS_MSE	Среднеквадратичная ошибка
LOSS_MAE	Средняя абсолютная ошибка
LOSS_CCE	Категориальная кроссэнтропия
LOSS_BCE	Бинарная кроссэнтропия
LOSS_MAPE	Средняя абсолютная ошибка в процентах
LOSS_MSLE	Среднеквадратическая логарифмическая ошибка
LOSS_KLD	Дивергенция Кульбака-Лейблера
LOSS_COSINE	Косинусное сходство/близость

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
LOSS_POISSON	Функция потерь Пуассона
LOSS_HINGE	Кусочно-линейная функция потерь (Hinge loss)
LOSS_SQ_HINGE	Квадратичная кусочно-линейная функция потерь
LOSS_CAT_HINGE	Категориальная кусочно-линейная функция потерь
LOSS_LOG_COSH	Логарифм гиперболического косинуса
LOSS_HUBER	Функция потерь Юбера

ENUM_ACTIVATION_FUNCTION

Перечисление для активационной функции vector::Activation и производной активационной функции vector::Derivative.

Идентификатор	Описание	Параметры
AF_NONE	Функция активации не используется, значение на входе передается на выход	
AF_ELU	Экспоненциальная линейная единица	
AF_EXP	Экспоненциальная	
AF_GELU	Линейная единица ошибки Гаусса	
AF_HARD_SIGMOID	Жесткий сигмоид	
AF_LINEAR	Линейная	
AF_LRELU	Линейный выпрямитель с "утечкой" (Leaky ReLU)	
AF_RELU	Усеченное линейное преобразование ReLU	
AF_SELU	Масштабированная экспоненциальная линейная функция (Scaled ELU)	
AF_SIGMOID	Сигмоид	
AF_SOFTMAX	Softmax	
AF_SOFTPLUS	Softplus	

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Параметры
AF_SOFTSIGN	Softsign	
AF_SWISH	Swish-функция	
AF_TANH	Гиперболический тангенс	
AF_TRELU	Линейный выпрямитель с порогом	

ENUM_SORT_MODE

Перечисление типов сортировки для функции Sort.

Идентификатор	Описание
SORT_ASCENDING	Сортировка по возрастанию
SORT_DESCENDING	Сортировка по убыванию

ENUM_MATRIX_AXIS

Перечисление для указания оси

Идентификатор	Описание
AXIS_NONE	Ось не задана
AXIS_HORZ	Горизонтальная ось
AXIS_VERT	Вертикальная ось

© 2000-2025, MetaQuotes Ltd.

Инициализация

Предусмотрено несколько способов объявления и инициализации матриц и векторов.

Функция	Действие
Assign	Копирует матрицу, вектор или массив с автоматическим преобразованием
CopyIndicatorBuffer	Получает в вектор данные указанного буфера указанного индикатора в указанном количестве
CopyRates	Получает в матрицу или вектор исторические серии структуры MqlRates указанного символа-периода в указанном количестве
CopyTicks	Получает в матрицу или вектор тики из структуры MqlTick
CopyTicksRange	Получает в матрицу или вектор тики из структуры MqlTick в указанном диапазоне дат
Eye	Возвращает матрицу с единицами по диагонали и нулями в остальных местах
Identity	Создает единичную матрицу указанного размера
Ones	Создает и возвращает новую матрицу, заполненную единицами
Zeros	Создает и возвращает новую матрицу, заполненную нулями
Full	Создает и возвращает новую матрицу, заполненную заданными значениями
Tri	Создает матрицу с единицами по указанной диагонали и ниже нее и нулями в остальных местах
Init	Инициализирует матрицу или вектор
Fill	Заполняет существующую матрицу или вектор заданным значением
Random	Статическая функция. Создает и возвращает новую матрицу или вектор, заполненный случайными значениями. Случайные значения генерируются равномерно в заданном диапазоне.

Объявление без указания размера (без распределения памяти для данных):

```
matrix         matrix_a;   // матрица типа double
matrix<double> matrix_a1;  // другой способ объявления матрицы типа double, подходит
matrixf        matrix_a2;  // матрица типа float
matrix<float>  matrix_a3;  // матрица типа float
vector         vector_a;   // вектор типа double
vector<double> vector_a1;
vectorf        vector_a2;  // вектор типа float
vector<float>  vector_a3;
```

© 2000-2025, MetaQuotes Ltd.

Объявление с указанием размера (с распределением памяти для данных, но без какой-либо инициализации):

```
  matrix         matrix_a(128,128);           // в качестве параметров можно указыват
  matrix<double> matrix_a1(InpRows,InpCols);  // так и переменные
  matrixf        matrix_a2(1,128);            // аналог горизонтального вектора
  matrix<float>  matrix_a3(InpRows,1);        // аналог вертикального вектора
  vector         vector_a(256);
  vector<double> vector_a1(InpSize);
  vectorf        vector_a2(SomeFunc());       // функция SomeFunc возвращает число тип
  vector<float>  vector_a3(InpSize+16);       // в качестве параметра можно использова
```

Объявление с инициализацией (размеры матриц и векторов в данном случае определяются инициализирующей последовательностью):

```
  matrix         matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
  matrix<double> matrix_a1=matrix_a;                    // должны быть матрицы одно
  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;                   // должны быть векторы одно
```

Объявление с инициализацией:

```
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); // сначала создаётся неиниц
  matrixf matrix_a1(10,20,MatrixArange);                  // после создания матрицы вы
  vector  vector_a(size,VectorArange,-10.0);              // после создания вектора вы
  vectorf vector_a1(128,VectorArange);
```

© 2000-2025, MetaQuotes Ltd.

Необходимо иметь в виду, что размеры матрицы или вектора могут быть изменены, так как память для данных всегда динамическая.

Статические методы

Статические методы создания матриц и векторов указанного размера с инициализацией определённым способом:

```
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);
```

Методы инициализации уже созданных матриц и векторов:

```
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();
```

© 2000-2025, MetaQuotes Ltd.

Assign

Копирует матрицу, вектор или массив с автоматическим преобразованием.

```
bool matrix::Assign(
  const matrix<T>  &mat     // копируемая матрица
   );
bool matrix::Assign(
  const void       &array[] // копируемый массив
   );
bool vector::Assign(
  const vector<T>  &vec     // копируемый вектор
   );
bool vector::Assign(
  const void       &array[] // копируемый массив
   );
```

Параметры

m, v или array

[in] Матрица, вектор или массив, откуда копируются значения.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

В отличие от Copy, метод Assign позволяет копировать еще и массивы. При этом происходит автоматическое приведение типа, а результирующая матрица или вектор подстраивается под размеры копируемого массива.

Пример:

```
//--- копирование матриц
  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);

//--- копирование массива в матрицу
  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);
```

© 2000-2025, 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]]

*/
```

Смотри также

Copy

© 2000-2025, MetaQuotes Ltd.

CopyIndicatorBuffer

Получает в вектор данные указанного буфера указанного индикатора в указанном количестве.

Данные в вектор будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под вектор. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle индикатора
   ulong     buffer_index,         // номер буфера индикатора
   ulong     start_pos,            // откуда начнем
   ulong     count                 // сколько копируем
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle индикатора
   ulong     buffer_index,         // номер буфера индикатора
   datetime  start_time,           // с какой даты
   ulong     count                 // сколько копируем
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle индикатора
   ulong     buffer_index,         // номер буфера индикатора
   datetime  start_time,           // с какой даты
   datetime  stop_time             // по какую дату
   );
```

Параметры

indicator_handle

[in] Хэндл индикатора, полученный соответствующей индикаторной функцией.

buffer_index

[in] Номер индикаторного буфера.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

© 2000-2025, MetaQuotes Ltd.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Отсчет элементов копируемых данных (индикаторный буфер с индексом buffer_index) от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар (значение индикатора для текущего бара).

При копировании заранее неизвестного количества данных желательно объявлять вектор без указания размера (без распределения памяти для данных), так как функция CopyBuffer() старается распределить размер принимающего вектора под размер копируемых данных.

Если необходимо произвести частичное копирование значений индикатора, то для этих целей необходимо использовать промежуточный вектор, в который копируется требуемое количество. И уже из этого вектора-посредника произвести поэлементное копирование нужного количества значений в нужные места принимающего вектора.

Если необходимо копировать заранее известное количество данных, то лучше для этого предварительно объявить вектор с указанием размера, чтобы избежать излишнего перевыделения памяти.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, то функция сразу же вернет false, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута.

Смотри также

CopyBuffer

© 2000-2025, MetaQuotes Ltd.

CopyRates

Получает в матрицу или вектор исторические серии структуры MqlRates указанного символа-периода в указанном количестве. Отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.

При этом данные будут скопированы таким образом, что самый старый по времени элемент помещается в начало матрицы/вектора. Существует 3 варианта метода.

Обращение по начальной позиции и количеству требуемых элементов

```
bool  matrix::CopyRates(
   string           symbol,       // имя символа
   ENUM_TIMEFRAMES  period,       // период
   ulong            rates_mask,   // комбинация флагов для указания запрашиваемых сери
   ulong            start,        // номер начального бара, с которого начнем копирова
   ulong            count         // сколько копируем
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
bool  matrix::CopyRates(
   string           symbol,       // имя символа
   ENUM_TIMEFRAMES  period,       // период
   ulong            rates_mask,   // комбинация флагов для указания запрашиваемых сери
   datetime         from,         // с какой даты
   ulong            count         // сколько копируем
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

```
bool  matrix::CopyRates(
   string           symbol,       // имя символа
   ENUM_TIMEFRAMES  period,       // период
   ulong            rates_mask,   // комбинация флагов для указания запрашиваемых сери
   datetime         from,         // с какой даты
   datetime         to            // по какую дату
   );
```

Для вектора методы аналогичны.

Обращение по начальной позиции и количеству требуемых элементов

```
bool  vector::CopyRates(
   string           symbol,       // имя символа
   ENUM_TIMEFRAMES  period,       // период
   ulong            rates_mask,   // комбинация флагов для указания запрашиваемых сери
   ulong            start,        // номер начального бара, с которого начнем копирова
   ulong            count         // сколько копируем
   );
```

Обращение по начальной дате и количеству требуемых элементов

© 2000-2025, MetaQuotes Ltd.

```
bool  vector::CopyRates(
   string           symbol,       // имя символа
   ENUM_TIMEFRAMES  period,       // период
   ulong            rates_mask,   // комбинация флагов для указания запрашиваемых сери
   datetime         from,         // с какой даты
   ulong            count         // сколько копируем
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

```
bool  vector::CopyRates(
   string           symbol,       // имя символа
   ENUM_TIMEFRAMES  period,       // период
   ulong            rates_mask,   // комбинация флагов для указания запрашиваемых сери
   datetime         from,         // с какой даты
   datetime         to            // по какую дату
   );
```

Параметры

symbol

[in] Символ.

period

[in] Период.

rates_mask

[in] Комбинация флагов из перечисления ENUM_COPY_RATES, указывающих тип запрашиваемых серий. При копировании в вектор можно указать только одно значение из перечисления ENUM_COPY_RATES, иначе будет ошибка.

start

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

from

[in] Время бара, соответствующее первому элементу.

to

[in] Время бара, соответствующее последнему элементу.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает false. В случае если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет false.

© 2000-2025, MetaQuotes Ltd.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и учитывается с точностью до секунды. То есть дата открытия любого бара, для которого возвращается значение (объем, спред, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

ENUM_COPY_RATES

Перечисление ENUM_COPY_RATES содержит флаги для указания типа данных, которые необходимо передать в матрицу или массив. Комбинация флагов позволяет за один запрос получить несколько серий из истории. При этом порядок строк в матрице будет соответствовать порядку значений в перечислении ENUM_COPY_RATES — то есть строка с данными High будет в матрице всегда выше строки с данными Low.

Идентификатор	Значение	Описание
COPY_RATES_OPEN	1	Серия цен Open
COPY_RATES_HIGH	2	Серия цен High
COPY_RATES_LOW	4	Серия цен Low
COPY_RATES_CLOSE	8	Серия цен Close
COPY_RATES_TIME	16	Серия Time (время открытия бара) Получение времени во float вектора и матрицы (vectord и matrixf), приводит к потерям ~100 секунд, т.к. точность float сильно ограничена, и целые числа со значением более 1<<24 не могут быть точно представлены во float.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение	Описание
COPY_RATES_VOLUME_TICK	32	Тиковые объемы
COPY_RATES_VOLUME_REAL	64	Торговые объемы
COPY_RATES_SPREAD	128	Спреды
Комбинация		
COPY_RATES_OHLC	15	Серии Open, High, Low и Close
COPY_RATES_OHLCT	31	Серии Open, High, Low, Close и Time
Расположение данных		
COPY_RATES_VERTICAL	32768	Серии копируются в матрицу вдоль вертикальной оси. Это означает, что полученные значения серии в матрице расположены вертикально — то есть самые старые данные находятся в первой строке, а самые свежие в последней строке матрицы. По умолчанию при копировании в матрицу серии добавляются вдоль горизонтальной оси. Флаг имеет значение только при копировании в матрицу.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- получим котировки в матрицу
  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());
//--- проверка
  MqlRates mql_rates[];
  if(CopyRates(Symbol(), PERIOD_CURRENT, 1, 10, mql_rates)>0)
   {
    Print("mql_rates array:");
    ArrayPrint(mql_rates);
   }
  else
    Print("CopyRates(Symbol(), PERIOD_CURRENT,1, 10, mql_rates). Error ", GetLastErro
```

© 2000-2025, MetaQuotes Ltd.

```
//--- получим котировки в вектор = неправильный вызов
  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());
//--- получим цены закрытия в вектор
  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(
    [0.9961100000000001,0.99491,0.99426,0.99441,0.99448,0.99494,0.9964499999999999,0.9
    [0.99641,0.99588,0.99441,0.99464,0.99594,0.99697,0.99758,0.99581,0.995280000000000(
    [1662436800,1662440400,1662444000,1662447600,1662451200,1662454800,1662458400,1662
   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,
*/
```

Смотри также

Доступ к таймсериям и индикаторам, CopyRates

© 2000-2025, MetaQuotes Ltd.

CopyTicks

Получает в матрицу или вектор тики из структуры MqlTick. Отсчет элементов от стартовой позиции ведется от прошлого к настоящему, то есть тик с индексом 0 является самым старым. Для анализа тика необходимо проверять поле *flags*, которое сообщает, что именно было изменено в данном тике.

```
bool  matrix::CopyTicks(
   string           symbol,              // имя символа
   ulong            flags,               // флаг, определяющий тип получаемых тиков
   ulong            from_msc,            // время, начиная с которого запрашиваются
   ulong            count                // количество тиков, которые необходимо по
   );
```

Метод вектора

```
bool  vector::CopyTicks(
   string           symbol,              // имя символа
   ulong            flags,               // флаг, определяющий тип получаемых тиков
   ulong            from_msc,            // время, начиная с которого запрашиваются
   ulong            count                // количество тиков, которые необходимо по
   );
```

Параметры

symbol

[in] Символ.

flags

[in] Комбинация флагов из перечисления ENUM_COPY_TICKS, указывающих состав запрашиваемых данных. При копировании в вектор можно указать только одно значение из перечисления ENUM_COPY_TICKS, иначе будет ошибка.

from_msc

[in] Время, начиная с которого запрашиваются тики. Указывается в миллисекундах с 01.01.1970. Если параметр from_msc=0, то отдаются последние count тиков.

count

[in] Количество запрашиваемых тиков. Если параметры from_msc и count не указаны, то будут записаны все доступные последние тики, но не более 2000.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Первый вызов CopyTicks() инициирует синхронизацию базы тиков, хранящихся на жёстком диске по данному символу. Если тиков в локальной базе не хватает, то недостающие тики автоматически будут загружены с торгового сервера. При этом будут синхронизированы тики с момента времени ***from_msc***, указанного в CopyTicks(), по текущий момент. После этого все

© 2000-2025, MetaQuotes Ltd.

приходящие по данному символу тики будут поступать в тиковую базу и поддерживать её в актуальном синхронизированном состоянии.

Если параметры *from_msc* и *count* не указаны, то в матрицу/вектор будут записаны все доступные тики, но не более 2000.

В индикаторах метод CopyTicks() возвращает результат немедленно: При вызове из индикатора CopyTick() сразу же вернёт доступные по символу тики, а также запустит синхронизацию базы тиков, если данных не хватило. Все индикаторы на одном символе работают в одном общем потоке, поэтому индикатор не имеет права ждать завершения синхронизации. После окончания синхронизации при последующем вызове CopyTicks() вернёт все запрашиваемые тики. Функция OnCalculate() в индикаторах вызывается после поступления каждого тика.

В экспертах и скриптах метод CopyTicks() может дожидаться результата до 45 секунд: В отличие от индикатора каждый эксперт и скрипт работает в собственном потоке, и поэтому может дожидаться окончания синхронизации до 45 секунд. Если за это время тики так и не будут синхронизированы в необходимом объёме, то CopyTicks() по таймауту вернёт только имеющиеся в наличии тики, при этом синхронизация продолжится. Функция OnTick() в экспертах не является обработчиком каждого тика, она лишь уведомляет эксперта об изменениях на рынке. Изменения могут быть пакетными: в терминал может одновременно прийти несколько тиков, но функция OnTick() будет вызвана лишь один раз для уведомления эксперта о последнем состоянии рынка.

Скорость выдачи: терминал хранит по каждому символу 4096 последних тиков в кеше для быстрого доступа (для символов с запущенным стаканом - 65536 тиков), запросы к этим данным выполняются быстрее всего. При запросе тиков текущей торговой сессии за пределами кеша CopyTicks() обращается уже к тикам, которые хранятся в памяти терминала, эти запросы требуют большего времени на выполнение. Самыми медленными являются запросы тиков за другие дни, так как в этом случае данные читаются уже с диска.

ENUM_COPY_TICKS

Перечисление ENUM_COPY_TICKS содержит флаги для указания типа данных, которые необходимо передать в матрицу или массив. Комбинация флагов позволяет за один запрос получить несколько серий из истории. При этом порядок строк в матрице будет соответствовать порядку значений в перечислении ENUM_COPY_TICKS — то есть строка с данными High будет в матрице всегда выше строки с данными Low.

Идентификатор	Значение	Описание
COPY_TICKS_INFO	1	Тики, содержащие изменения цен Bid и/или Ask
COPY_TICKS_TRADE	2	Тики, содержащие изменения цены Last и/или объема (Volume)
COPY_TICKS_ALL	3	Все тики
COPY_TICKS_TIME_MS	1<<8	Время тика в миллисекундах
COPY_TICKS_BID	1<<9	Цена Bid
COPY_TICKS_ASK	1<<10	Цена Ask
COPY_TICKS_LAST	1<<11	Цена Last (цена последней сделки)

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение	Описание
COPY_TICKS_VOLUME	1<<12	Объем для цены Last
COPY_TICKS_FLAGS	1<<13	Флаги тиков
Расположение данных		
COPY_TICKS_VERTICAL	1<<15	Тики копируются в матрицу вдоль вертикальной оси. Это означает, что полученные тики в матрице расположены вертикально — то есть самые старые тики находятся в первой строке, а самые свежие в последней строке матрицы. По умолчанию при копировании в матрицу тики добавляются вдоль горизонтальной оси. Флаг имеет значение только при копировании в матрицу.

Чтобы узнать, какие именно данные изменились с текущим тиком, анализируйте его флаги:

- TICK_FLAG_BID - тик изменил цену бид
- TICK_FLAG_ASK - тик изменил цену аск
- TICK_FLAG_LAST - тик изменил цену последней сделки
- TICK_FLAG_VOLUME - тик изменил объем
- TICK_FLAG_BUY - тик возник в результате сделки на покупку
- TICK_FLAG_SELL - тик возник в результате сделки на продажу

Смотри также

Доступ к таймсериям и индикаторам, CopyTicks

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange

Получает в матрицу или вектор тики из структуры MqlTick в указанном диапазоне дат. Отсчет элементов от стартовой позиции ведется от прошлого к настоящему, то есть тик с индексом 0 является самым старым. Для анализа тика необходимо проверять поле *flags*, которое сообщает, что именно было изменено в данном тике.

```
bool  matrix::CopyTicksRange(
   string           symbol,                 // имя символа
   ulong            flags,                  // флаг, определяющий тип получаемых тиков
   ulong            from_msc,               // время, начиная с которого запрашиваются
   ulong            to_msc                  // время, по которое запрашиваются тики
   );
```

Метод вектора

```
bool  vector::CopyTicksRange(
   string           symbol,                 // имя символа
   ulong            flags,                  // флаг, определяющий тип получаемых тиков
   ulong            from_msc,               // время, начиная с которого запрашиваются
   ulong            to_msc                  // время, по которое запрашиваются тики
   );
```

Параметры

symbol

[in] Символ.

flags

[in] Комбинация флагов из перечисления ENUM_COPY_TICKS, указывающих состав запрашиваемых данных. При копировании в вектор можно указать только одно значение из перечисления ENUM_COPY_TICKS, иначе будет ошибка.

from_msc

[in] Время, начиная с которого запрашиваются тики. Указывается в миллисекундах с 01.01.1970. Если параметр from_msc не указан, то отдаются тики от самого начала истории. Отдаются тики со временем >= from_msc.

to_msc

[in] Время, по которое запрашиваются тики. Указывается в миллисекундах с 01.01.1970. Отдаются тики со временем <= to_msc. Если параметр to_msc не указан, то отдаются все тики до конца истории.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки. GetLastError() может возвращать следующие ошибки:

- ERR_HISTORY_TIMEOUT - время ожидание синхронизации тиков вышло, функция отдала всё что было.
- ERR_HISTORY_SMALL_BUFFER - статический буфер слишком маленький, отдано столько, сколько поместилось в массив.

© 2000-2025, MetaQuotes Ltd.

- ERR_NOT_ENOUGH_MEMORY – не хватает памяти для получения истории из указаного диапазона в динамический массив тиков. Не удалось выделить нужный объем памяти под массиве тиков.

Чтобы узнать, какие именно данные изменились с текущим тиком, анализируйте его флаги:

- TICK_FLAG_BID – тик изменил цену бид
- TICK_FLAG_ASK – тик изменил цену аск
- TICK_FLAG_LAST – тик изменил цену последней сделки
- TICK_FLAG_VOLUME – тик изменил объем
- TICK_FLAG_BUY – тик возник в результате сделки на покупку
- TICK_FLAG_SELL – тик возник в результате сделки на продажу

Примечание

Метод CopyTicksRange() предназначен для запроса тиков из строго указанного диапазона, например, за конкретный день истории. В то время как CopyTicks() позволяет указать только начальную дату, например – получить все тики с начала месяца по текущий момент.

Смотри также

Доступ к таймсериям и индикаторам, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

Eye

Статическая функция создаёт матрицу указанного размера с единицами в указанной диагонали и нулями за пределами диагонали. Возвращает матрицу с единицами по диагонали и нулями в остальных местах.

```
static matrix matrix::Eye(
  const ulong  rows,         // количество строк
  const ulong  cols,         // количество столбцов
  const int    ndiag=0       // индекс диагонали
   );
```

Параметры

rows

[in] Количество строк на выходе.

cols

[in] Количество столбцов на выходе.

ndiag=0

[in] Индекс диагонали: 0 (по умолчанию) означает главную диагональ, положительное значение означает диагональ выше, а отрицательное — ниже.

Возвращаемое значение

Матрица, в которой все элементы равны нулю, кроме k-й диагонали, значения которой равны единице.

Пример на 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]]
  */
```

Пример на Python:

```
np.eye(3, dtype=int)
```

© 2000-2025, 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.]])
```

© 2000-2025, MetaQuotes Ltd.

Identity

Статическая функция создаёт единичную матрицу указанного размера (необязательно квадратную). Единичная матрица содержит единицы в главной диагонали и нули за пределами главной диагонали. Главная диагональ состоит из элементов матрицы, индексы строки и столбца которого равны, то есть [0,0],[1,1],[2,2] и т.д. Создает новую единичную матрицу.

Также существует метод Identity, который преобразует уже существующую матрицу в единичную.

```
static matrix matrix::Identity(
  const ulong  rows,         // количество строк
  const ulong  cols,         // количество столбцов
   );

void matrix::Identity();
```

Параметры

rows

[in] Количество строк (и столбцов) в матрице n x n.

Возвращаемое значение

Возвращает единичную матрицу. Единичная матрица — квадратную матрицу с единицами на главной диагонали.

Пример на 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]]
*/
```

Пример на Python:

```
np.identity(3)
```

© 2000-2025, MetaQuotes Ltd.

```
array([[1.,  0.,  0.],
       [0.,  1.,  0.],
       [0.,  0.,  1.]])
```

© 2000-2025, MetaQuotes Ltd.

Ones

Статическая функция. Создает и возвращает новую матрицу, заполненную единицами.

```
static matrix matrix::Ones(
  const ulong  rows,      // количество строк
  const ulong  cols       // количество столбцов
   );

static vector vector::Ones(
  const ulong  size,      // размер вектора
   );
```

Параметры

rows

[in] Количество строк.

cols

[in] Количество столбцов.

Возвращаемое значение

Возвращает новую матрицу из заданных строк и столбцов, заполненную единицами.

Пример на 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]]
*/
```

Пример на Python:

```
np.ones((4, 1))
array([[1.],
       [1.]])
```

© 2000-2025, MetaQuotes Ltd.

Zeros

Статическая функция. Создает и возвращает новую матрицу, заполненную нулями.

```
static matrix matrix::Zeros(
  const ulong  rows,      // количество строк
  const ulong  cols       // количество столбцов
   );

static vector vector::Zeros(
  const ulong  size,      // размер вектора
   );
```

Параметры

rows

[in] Количество строк.

cols

[in] Количество столбцов.

Возвращаемое значение

Возвращает новую матрицу из заданных строк и столбцов, заполненную нулями.

Пример на MQL5:

```
  matrix zeros=matrix::Zeros(3, 4);
  Print("zeros = \n", zeros);
/*
zeros =
   [[0,0,0,0]
    [0,0,0,0]
    [0,0,0,0]]
*/
```

Пример на Python:

```
np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

© 2000-2025, MetaQuotes Ltd.

Full

Статическая функция. Создает и возвращает новую матрицу, заполненную указанным значением.

```
static matrix matrix::Full(
  const ulong   rows,       // количество строк
  const ulong   cols,       // количество столбцов
  const double  value       // значение для заполнения
   );

static vector vector::Full(
  const ulong   size,       // размер вектора
  const double  value       // значение для заполнения
   );
```

Параметры

rows

[in] Количество строк.

cols

[in] Количество столбцов.

value

[in] Значение, которым заполнятся все элементы матрицы.

Возвращаемое значение

Возвращает новую матрицу из заданных строк и столбцов, заполненную указанным значением.

Пример на 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]]
*/
```

Пример:

```
np.full((2, 2), 10)
array([[10, 10],
       [10, 10]])
```

© 2000-2025, MetaQuotes Ltd.

Tri

Статическая функция. Создает матрицу с единицами по указанной диагонали и ниже нее и нулями в остальных местах.

```
static matrix matrix::Tri(
  const ulong rows,         // количество строк
  const ulong cols,         // количество столбцов
  const int   ndiag=0       // количество диагоналей
   );
```

Параметры

rows

[in] Количество строк в матрице.

cols

[in] Количество столбцов в матрице.

ndiag=0

[in] Поддиагональ матрицы, которая и остальные ниже которой будут заполнены. k = 0 — главная диагональ, k < 0 — диагонали ниже главной, k > 0 — выше. По умолчанию - 0.

Возвращаемое значение

Матрица, нижний треугольник которой заполнен единицами, а все остальные места — нулями.

Пример на 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]]
*/
```

Пример:

```
np.tri(3, 5, 2, dtype=int)
array([[1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0],
       [1, 1, 1, 1, 1]])
```

© 2000-2025, MetaQuotes Ltd.

Init

Инициализирует матрицу или вектор.

```
void matrix::Init(
  const ulong  rows,            // количество строк
  const ulong  cols,            // количество столбцов
   func_name   init_func=NULL,  // функция инициализации, помещенная в какую-либо обла
   ...         parameters
   );

void vector::Init(
  const ulong  size,            // размер вектора
  func_name    init_func=NULL,  // функция инициализации, помещенная в какую-либо обла
   ...         parameters
   );
```

Параметры

rows

[in] Количество строк.

cols

[in] Количество столбцов.

func_name

[in] Инициализирующая функция.

...

[in] Параметры инициализирующей функции.

Возвращаемое значение

Нет возвращаемого значения.

Пример:

```
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)
  {
```

© 2000-2025, 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); // сначала создаётся неинициализирова
  Print("matrix m \n",m);                        // затем вызывается функция MatrixAra
  matrixf m_float(5,5,MatrixArange,-2.f,0.1f);   // после создания матрицы типа float
  Print("matrix m_float \n",m_float);
  vector  v(size_k,VectorArange,-10.0);          // после создания вектора вызывается
  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,-0.9999999999999992,-0.8999999999999992]]
   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]
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

Fill

Заполняет существующую матрицу или вектор заданным значением.

```
void matrix::Fill(
  const double  value      // значение для заполнения
   );

void vector::Fill(
  const double  value      // значение для заполнения
   );
```

Параметры

value

[in] Значение, которым заполнятся все элементы матрицы.

Возвращаемое значение

Нет возвращаемого значения. Матрица заполняется по месту указанным значением.

Пример:

```
   matrix matrix_a(2,2);
   matrix_a.Fill(10);
   Print("matrix_a\n",matrix_a);

/*
  matrix_a
  [[10,10]
   [10,10]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Random

Статическая функция. Создает и возвращает новую матрицу или вектор, заполненный случайными значениями. Случайные значения генерируются равномерно в заданном диапазоне.

```
static vector vector::Random(
  const ulong   size,        // длина вектора
  const double  min=0.0,     // минимальное значение
  const double  max=1.0      // максимальное значение
   );

static matrix matrix::Random(
  const ulong   rows,        // количество строк
  const ulong   cols         // количество столбцов
  const float   min=0.0,     // минимальное значение
  const float   max=1.0      // максимальное значение
   );
```

Функция для заполнения существующей матрицы или вектора случайными значениями. Случайные значения генерируются равномерно в заданном диапазоне.

```
void  vector::Random(
  const double  min=0.0,     // минимальное значение
  const double  max=1.0      // максимальное значение
   );

void  matrix::Random(
  const float  min=0.0,      // минимальное значение
  const float  max=1.0       // максимальное значение
   );
```

Параметры

rows

[in] Количество строк.

cols

[in] Количество столбцов.

size

[in] Длина вектора.

min=0.0

[in] Минимальное значение в генерируемой выборке случайных чисел.

max=1.0

[in] Максимальное значение в генерируемой выборке случайных чисел.

Возвращаемое значение

Возвращает новую матрицу из заданных строк и столбцов, случайными значениями.

© 2000-2025, MetaQuotes Ltd.

Пример

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector v1= vector::Random(6,0,4);
   Print("vector v1 \n", v1);
   /*
   vector v1
   [3.340834286841758,0.9073984578501895,3.15598511117417,0.7106475841045956,2.294010(
   */

   vector v2(4);
   v2.Random(-3,3);
   Print("vector v2 \n", v2);
   /*
   vector v2
   [-2.642032080246609,1.008521607147651,1.340623848219547,-2.789016363161853]
   */

   matrix m1 = matrix::Random(5, 5, -1, 3);
   Print("matrix m1 \n", m1);
   /*
   matrix m1
   [[1.468433369525809,1.199383023707052,0.6154667729547869,2.830012157519816,-0.06995
    [2.638591175814763,2.966658948912904,2.001837257994767,-0.0857080246987073,0.826(
    [0.982665140643405,0.5825661973987162,-0.6615881071569371,0.4069533109215444,-0.16
    [-0.5791836119208447,-0.9332020478167575,2.801898707589577,-0.09887318478405316,-(
    [-0.1069569017521471,-0.04576879345014895,0.6320100840255642,-0.3805729270401813,(
   */

   matrix m2(3, 3);
   m2.Random(-2, 2);
   Print("matrix m2 \n", m2);
   /*
   matrix m2
   [[-1.928191291151613,-1.222873445419332,-1.965333905062949]
    [-1.077723600686929,1.053317863273741,-0.1786824246353196]
    [-1.965241401448204,1.620956906029246,-1.696386492740453]]
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Манипуляции над матрицами и векторами

Методы для проведения базовых операций над матрицами — заполнение, копирование, получение части матрицы, транспонирование, разделение и сортировка.

Есть также несколько методов для работы со строками и столбцами матрицы.

Функция	Действие
HasNan	Возвращает количество NaN значений в матрице/векторе
Transpose	Разворачивает или переставляет оси матрицы, возвращает измененную матрицу
TransposeConjugate	Транспонирование комплексной матрицы с сопряжением. Разворачивает или переставляет оси матрицы с изменением знака мнимой части комплексного числа, возвращает измененную матрицу
TriL	Возвращает копию матрицы, заменив все элементы выше k-й диагонали на нули. Нижнетреугольная матрица
TriU	Возвращает копию матрицы, заменив все элементы ниже k-й диагонали на нули. Верхнетреугольная матрица
Diag	Извлекает диагональ или создает диагональную матрицу
Row	Возвращает вектор-строку. Записывает вектор в указанную строку
Col	Возвращает вектор-столбец. Записывает вектор в указанный столбец
Copy	Возвращает копию данной матрицы/вектора
Concat	Объединяет 2 подматрицы в одну матрицу. Объединяет 2 вектора в один
Compare	Сравнивает элементы двух матриц/векторов с указанной точностью
CompareByDigits	Сравнивает элементы двух матриц/векторов на совпадение с точностью до значащих цифр
CompareEqual	Выполняет абсолютное сравнение двух матриц путем разворачивания последовательных строк в одномерные векторы
Flat	Позволяет адресоваться к элементу матрицы через один индекс, а не через два
Clip	Ограничивает элементы матрицы/вектора указанным интервалом допустимых значений
Reshape	Изменяет форму матрицы без изменения его данных
Resize	Возвращает новую матрицу указанной формы и размера
SwapRows	Меняет местами строки в матрице

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
SwapCols	Меняет местами столбцы в матрице
Split	Разделение матрицы на несколько подматриц
Hsplit	Горизонтальное разделение матрицы на несколько подматриц. То же самое, что Split с axis=0
Vsplit	Вертикальное разделение матрицы на несколько подматриц. То же самое, что Split с axis=1
ArgSort	Производит косвенную сортировка матрицы или вектора.
Sort	Производит сортировку матрицы или вектора по месту.

© 2000-2025, MetaQuotes Ltd.

HasNan

Возвращает количество NaN значений в матрице/векторе.

```
ulong vector::HasNan();

ulong matrix::HasNan();
```

Возвращаемое значение

Количество элементов матрицы/вектора, которые содержат значение NaN.

Примечание

Методы Compare и CompareByDigits при сравнении соответствующей пары элементов, имеющих значения NaN, считают эти элементы равными, тогда как при обычном сравнении чисел с плавающей точкой NaN != NaN.

Пример:

```
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);
  }

/* Результат:

 single: false
 vector: true
*/
```

Смотри также

MathClassify, Compare, CompareByDigits

© 2000-2025, MetaQuotes Ltd.

Transpose

Транспонирование матрицы. Разворачивает или переставляет оси матрицы, возвращает измененную матрицу.

```
matrix matrix::Transpose()
```

Возвращаемое значение

Транспонированная матрица.

Простой алгоритм транспонирования матрицы на 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);
  }
```

Пример на 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]]
  */
```

Пример на Python:

```
import numpy as np

a = np.arange(6).reshape((2,3))
print("a \n",a)
print("np.transpose(a) \n",np.transpose(a))
```

© 2000-2025, MetaQuotes Ltd.

```
a
 [[0 1 2]
 [3 4 5]]
np.transpose(a)
 [[0 3]
 [1 4]
 [2 5]]
```

© 2000-2025, MetaQuotes Ltd.

TransposeConjugate

Транспонирование комплексной матрицы с сопряжением. Разворачивает или переставляет оси матрицы с изменением знака мнимой части комплексного числа, возвращает измененную матрицу.

```
matrix matrix::TransposeConjugate()
```

Возвращаемое значение

Транспонированная комплексно-сопряжённая матрица.

Простой алгоритм транспонирования комплексной матрицы с сопряжением — объяснение метода

```
//--- Функция транспонирования комплексной матрицы с сопряжением
matrix MatrixTranspose(const matrix& matrix_a) {
    //--- создаем новую матрицу matrix_c с размерами, обратными матрице matrix_a
    matrix matrix_c(matrix_a.Cols(), matrix_a.Rows());

    //--- проходим по всем строкам новой матрицы
    for (ulong i = 0; i < matrix_c.Rows(); i++) {
        //--- проходим по всем столбцам новой матрицы
        for (ulong j = 0; j < matrix_c.Cols(); j++) {
            //--- переносим вещественную часть элемента, транспонируя индекс
            matrix_c[i][j].real = matrix_a[j][i].real;

            //--- переносим мнимую часть элемента, меняя знак (сопряжение)
            matrix_c[i][j].imag = -matrix_a[j][i].imag;
        }
    }

    //--- возвращаем транспонированную с сопряжением матрицу
    return (matrix_c);
}
```

© 2000-2025, MetaQuotes Ltd.

TriL

Возвращает копию матрицы, заменив все элементы выше k-й диагонали на нули. Нижнетреугольная матрица.

```
matrix matrix::Tril(
  const int      ndiag=0      // индекс диагонали
   );
```

Параметры

ndiag=0

[in] Диагональ, выше которой элементы заменяются на нули. ndiag = 0 (по умолчанию) — основная диагональ, ndiag < 0 — ниже, а ndiag > 0 — выше основной диагонали.

Возвращаемое значение

Матрица, нижний треугольник которой заполнен единицами, а все остальные места — нулями.

Пример на 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]]
  */
```

Пример на 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]]
```

© 2000-2025, MetaQuotes Ltd.

TriU

Возвращает копию матрицы, заменив все элементы ниже k-й диагонали на нули. Верхнетреугольная матрица.

```
matrix matrix::Triu(
  const int      ndiag=0      // индекс диагонали
   );
```

Параметры

ndiag=0

[in] Диагональ, ниже которой элементы заменяются на нули. ndiag = 0 (по умолчанию) — основная диагональ, ndiag < 0 — ниже, а ndiag > 0 — выше основной диагонали.

Пример на 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]]
  */
```

Пример на 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]]
```

© 2000-2025, MetaQuotes Ltd.

Diag

Извлекает диагональ или создает диагональную матрицу.

```
vector matrix::Diag(
  const int      ndiag=0       // номер диагонали
   );

void matrix::Diag(
  const vector   v,            // диагональный вектор
  const int      ndiag=0       // номер диагонали
   );
```

Параметры

v

[in] Вектор, элементы которого должны быть вписаны в соответствующую диагональ (ndiag=0 — главная диагональ).

ndiag=0

[in] Диагональ. По умолчанию - 0. Для диагоналей выше основной указывается ndiag>0, для диагоналей ниже — ndiag<0.

Примечание

Устанавливать диагональ можно для нераспределённых матриц (не имеющих размеров). В этом случае будет создана нулевая матрица соответствующего диагональному вектору размера, после этого в соответствующей диагонали проставляются значения элементов вектора. Если диагональ устанавливается в уже существующую матрицу, то размеры матрицы не меняются и значения элементов матрицы за пределами вектора-диагонали не меняются.

Пример:

```
   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));
```

© 2000-2025, 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]
*/
```

© 2000-2025, MetaQuotes Ltd.

Row

Возвращает вектор-строку. Записывает вектор в указанную строку.

```
vector matrix::Row(
  const ulong   nrow      // номер строки
   );

void matrix::Row(
  const vector  v,        // вектор строка
  const ulong   nrow      // номер строки
   );
```

Параметры

nrow

[in] Номер строки.

Возвращаемое значение

Вектор.

Примечание

Устанавливать строку можно для нераспределённых матриц (не имеющих размеров). В этом случае будет создана нулевая матрица размером номер строки+1 x размер вектора, после этого в соответствующей строке проставляются значения элементов вектора. Если строка устанавливается в уже существующую матрицу, то размеры матрицы не меняются и значения элементов матрицы за пределами вектора-строки не меняются.

Пример:

```
   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]
```

© 2000-2025, 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]
   */
```

© 2000-2025, MetaQuotes Ltd.

Col

Возвращает вектор-столбец. Записывает вектор в указанный столбец.

```
vector matrix::Col(
  const ulong   ncol      // номер столбца
   );

void matrix::Col(
  const vector  v,        // вектор столбец
  const ulong   ncol      // номер столбца
   );
```

Параметры

ncol

[in] Номер столбца.

Возвращаемое значение

Вектор.

Примечание

Устанавливать столбец можно для нераспределённых матриц (не имеющих размеров). В этом случае будет создана нулевая матрица размером размер вектора x номер столбца+1, после этого в соответствующем столбце проставляются значения элементов вектора. Если столбец устанавливается в уже существующую матрицу, то размеры матрицы не меняются и значения элементов матрицы за пределами вектора-столбца не меняются.

Пример:

```
   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]
```

© 2000-2025, 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]
   */
```

© 2000-2025, MetaQuotes Ltd.

Copy

Создает копию данной матрицы/вектора.

```
bool matrix::Copy(
  const matrix&  a      // копируемая матрица
   );
bool vector::Copy(
  const vector&  v      // копируемый вектор
   );
```

Параметры

v

[in] Копируемая матрица или вектор.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример на 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]]
  */
  */
```

Пример на Python:

```
import numpy as np

a = np.eye(3,4)
print('a \n',a)
b = a
```

© 2000-2025, 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.]]
```

Смотри также

Assign

© 2000-2025, MetaQuotes Ltd.

Concat

Объединяет 2 подматрицы в одну матрицу. Объединяет 2 вектора в один

```
vector vector::Concat(
  const vector&  second_part      // второй вектор для объединения
   );

vector matrix::Concat(
  const matrix&  second_part      // вторая матрица для объединения
   );

matrix matrix::Split(
  const matrix&  second_part,     // вторая матрица для объединения
  const int      axis             // ось
   );
```

Параметры

second_part

[in] Второй вектор или матрица для объединения. Если матрицы объединены вдоль какой-либо из осей, то размеры матриц должны быть согласованы по этой оси.

axis

[in] Ось. 0 - горизонтальная ось, 1 - вертикальная ось.

Возвращаемое значение

Возвращает вектор, если векторы или матрицы объединены без параметра оси, или матрицы объединены вдоль горизонтальной или вертикальной оси.

Пример

```
   vector vector_a={1,2,3,4};
   vector vector_b={5,6,7};
   vector vector_c=vector_a.Concat(vector_b);
   Print("vector_c=",vector_c);

   matrix matrix_a={{1,2,3},{4,5,6}};
   matrix matrix_b={{7,8,9},{10,11,12}};
   vector_c=matrix_a.Concat(matrix_b);
   Print("vector_c=",vector_c);

   matrix matrix_c0=matrix_a.Concat(matrix_b,0);
   Print("matrix_c0=\n",matrix_c0);

   matrix matrix_c1=matrix_a.Concat(matrix_b,1);
   Print("matrix_c1=\n",matrix_c1);

  /*
```

© 2000-2025, MetaQuotes Ltd.

```
vector_c=[1,2,3,4,5,6,7]
vector_c=[1,2,3,4,5,6,7,8,9,10,11,12]
matrix_c0=
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
matrix_c1=
[[1,2,3,7,8,9]
 [4,5,6,10,11,12]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Compare

Сравнивает элементы двух матриц/векторов с указанной точностью.

```
ulong vector::Compare(
  const vector& vec,          // вектор для сравнения
  const double  epsilon       // точность
   );

ulong matrix::Compare(
  const matrix& mat,          // матрица для сравнения
  const double  epsilon       // точность
   );
```

Параметры

vec

[in] Вектор для сравнения.

mat

[in] Матрица для сравнения.

epsilon

[in] Точность.

Возвращаемое значение

Возвращает количество несовпавших элементов у сравниваемых матриц или векторов, то есть 0, если матрицы равны, больше 0, в противном случае.

Примечание

Операции сравнения == или != производят точное поэлементное сравнение. Как известно, точное сравнение вещественных чисел имеет ограниченное применение, поэтому был добавлен метод сравнения с эпсилоном. Бывают случаи, когда в пределах одной матрицы существуют элементы в диапазоне, например, от 1e-20 до 1e+20. Для этого предусмотрено поэлементное сравнение с учётом значащих цифр.

Для комплексных матриц/векторов сравнение заключается в оценке расстояния между комплексными числами. Расстояние рассчитывается по следующей формуле sqrt(pow(r1-r2, 2) + pow(i1-i2, 2) и является действительным числом, которое уже можно сравнивать с эпсилоном.

Пример:

```
   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);
```

© 2000-2025, MetaQuotes Ltd.

```
 ulong errors=matrix_check.Compare(matrix::Identity(3,3),1e-15);
 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

*/
```

© 2000-2025, MetaQuotes Ltd.

CompareByDigits

Сравнивает элементы двух матриц/векторов на совпадение с точностью значащих цифр.

```
ulong vector::CompareByDigits(
  const vector&  vec,          // вектор для сравнения
  const int      digits        // количество значащих цифр
   );

ulong matrix::CompareByDigits(
  const matrix&  mat,          // матрица для сравнения
  const int      digits        // количество значащих цифр
   );
```

Параметры

vec

[in] Вектор для сравнения.

mat

[in] Матрица для сравнения.

digits

[in] Количество значащих цифр для сравнения.

Возвращаемое значение

Возвращает количество несовпавших элементов у сравниваемых матриц или векторов, то есть 0, если матрицы равны, в противном случае значение больше 0.

Примечание

Операции сравнения == или != производят точное поэлементное сравнение. Как известно, точное сравнение вещественных чисел имеет ограниченное применение, поэтому был добавлен метод сравнения с эпсилоном. Бывают случаи, когда в пределах одной матрицы существуют элементы в диапазоне, например, от 1e-20 до 1e+20. Для этого предусмотрено поэлементное сравнение с учётом значащих цифр.

Пример:

```
   int    size_m=128;
   int    size_k=256;
   matrix matrix_a(size_m,size_k);
//--- заполним матрицу
   double  value=0.0;
   for(int i=0; i<size_m; i++)
     {
      for(int j=0; j<size_k; j++)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         if(i==j)
            matrix_a[i][j]=1.0+i;
         else
           {
            value+=1.0;
            matrix_a[i][j]=value/1e+20;
           }
        }
     }
//--- получим другую матрицу
   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
  */
```

© 2000-2025, MetaQuotes Ltd.

CompareEqual

Выполняет абсолютное сравнение двух матриц путем разворачивания последовательных строк в одномерные векторы.

```
ulong vector::Compare(
  const vector& vec    // вектор для сравнения
   );

ulong matrix::CompareEqual(
  const matrix& mat     // матрица для сравнения
   );
```

Параметры

vec

[in] Вектор для сравнения.

mat

[in] Матрица для сравнения.

Описание работы метода

Пусть у нас есть две матрицы: матрица A — матрица, для которой вызывается метод, и матрица B, которая передается в параметре метода. Сравнение выполняется следующим образом:

1. Матрицы разворачиваются в одномерные векторы путем последовательного объединения строк.
2. Векторы сравниваются поэлементно до первого несовпадающего элемента.
3. В зависимости от результатов сравнения возвращается одно из значений, описанных ниже.

Возвращаемое значение

-1 — если элемент матрицы A меньше соответствующего элемента матрицы B.

0 — если все элементы матриц A и B идентичны.

1 — если элемент матрицы A больше соответствующего элемента матрицы B.

Примечание

При сравнении учитываются элементы со значением NaN.

Элементы со значением NaN считаются равными, если они присутствуют в обеих матрицах на соответствующих позициях.

Знак NaN не учитывается при сравнении.

Элемент со значением NaN считается меньше любого другого числового значения.

Example

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

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

//---  абсолютное сравнение матриц
   matrix<double> A = matrix_a;  // Инициализация матрицы A
   matrix<double> B = matrix_c;  // Инициализация матрицы B
   int result = A.CompareEqual(B);
   switch(result)
     {
      case -1:
         Print("Матрица A меньше матрицы B");
         break;
      case 0:
         Print("Матрицы A и B идентичны");
         break;
      case 1:
         Print("Матрица A больше матрицы B");
         break;
      case -2:
         Print("Error! Матрица A неинициализирована");
         break;
      case 2:
         Print("Error! Матрица B неинициализирована");
         break;
      case -3:
         Print("Error! Размер матрицы A меньше размера матрицы B");
         break;
      case 3:
         Print("Error! Размер матрицы A больше размера матрицы B");
         break;
      default:
```

© 2000-2025, MetaQuotes Ltd.

```
         Print("Error! Неизвестная ошибка");
         break;
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

Flat

Позволяет адресоваться к элементу матрицы через один индекс, а не через два.

```
bool matrix::Flat(
  const ulong    index,      //
  const double    value      // устанавливаемое значение
   );

double matrix::Flat(
  const ulong    index,      //
   );
```

Параметры

index

[in] Flat-индекс

value

[in] Значение для установки по заданному индексу.

Возвращаемое значение

Значение по заданному индексу.

Примечание

Для матрицы matrix mat(3,3) доступы можно записать так:

- на чтение — x=mat.Flat(4), что эквивалентно записи x=mat[1][1]
- на запись — mat.Flat(5, 42), что эквивалентно записи mat[1][2]=42

Пример:

```
   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]]
```

© 2000-2025, MetaQuotes Ltd.

```
   max_value=12.0
   max_value=11.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Clip

Ограничивает элементы матрицы/вектора указанным интервалом допустимых значений.

```
bool matrix::Clip(
  const double  min_value,      // минимальное значение
  const double  max_value       // максимальное значение
   );
bool vector::Clip(
  const double  min_value,      // минимальное значение
  const double  max_value       // максимальное значение
   );
```

Параметры

min_value

[in] Минимальное значение.

max_value

[in] Максимальное значение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Матрица (или вектор) обрабатывается по месту. Никаких копий не создаётся.

Пример:

```
   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]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Reshape

Изменяет форму матрицы без изменения его данных.

```
void  Reshape(
  const ulong  rows,      // новое количество строк
  const ulong  cols       // новое количество столбцов
   );
```

Параметры

rows

[in] Новое количество строк.

cols

[in] Новое количество столбцов.

Примечание

Матрица обрабатывается по месту. Никаких копий не создаётся. Можно задавать любые размеры, то есть rows_new*cols_new!=rows_old*cols_old. При увеличении буфера матрицы "лишние" значения неопределены.

Пример:

```
   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]
```

© 2000-2025, MetaQuotes Ltd.

```
   [5,6,7,8]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Resize

Возвращает новую матрицу указанной формы и размера.

```
bool matrix::Resize(
  const ulong  rows,     //
  const ulong  cols,     // новое количество столбцов
  const ulong  reserve=0 // зарезервированное количество элементов
   );

bool vector::Resize(
  const ulong  size,     // новый размер
  const ulong  reserve=0 // зарезервированное количество элементов
   );
```

Параметры

rows

[in] Новое количество строк.

cols

[in] Новое количество столбцов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Матрица (или вектор) обрабатывается по месту. Никаких копий не создаётся. Можно задавать любые размеры, то есть rows_new*cols_new!=rows_old*cols_old. В отличие от Reshape матрица обрабатывается построчно. Если количество столбцов увеличивается, то значения "лишних" столбцов не определены. При увеличении количества строк значения элементов новых строк не определены. При уменьшении количества столбцов, каждая строка матрицы усекается.

Пример:

```
   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]
```

© 2000-2025, 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]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Set

Устанавливает значение для элемента вектора по указанному индексу .

```
bool vector::Set(
  ulong   index,      // индекс элемента
  double  value       // значение
   );
```

Параметры

index

[in] Индекс элемента, которому необходимо установить значение.

value

[in] Значение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Метод Set делает то же самое, что и присваивание значения через квадратные скобки, а именно: *vector[index]=value*. Добавлен для удобства переноса кода из языков, где используется именно такая запись. Пример ниже показывает оба варианта заполнения вектора значениями по указанному индексу.

Пример:

```
void OnStart()
  {
//---
   vector v1(10, VectorAssignValues);
   Print("v1 = ", v1);

   vector v2(10, VectorSetValues);
   Print("v2 = ", v2);
  }
  /* Результат
  v1 = [1,2,4,8,16,32,64,128,256,512]
  v2 = [1,2,4,8,16,32,64,128,256,512]
  */
//+------------------------------------------------------------------+
//| Заполняет вектор степенями числа через операцию присваивания     |
//+------------------------------------------------------------------+
void VectorAssignValues(vector& v, double initial=1)
  {
   double value=initial;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(ulong k=0; k<v.Size(); k++)
     {
      v[k]=value;
      value*=2;
     }
  }
//+------------------------------------------------------------------+
//| Заполняет вектор степенями числа методом 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;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SwapRows

Меняет местами строки в матрице.

```
bool matrix::SwapRows(
  const ulong  row1,     // индекс первой строки
  const ulong  row2      // индекс второй строки
   );
```

Параметры

row1

[in] Индекс первой строки.

row2

[in] Индекс второй строки.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
   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]]
  */
```

© 2000-2025, MetaQuotes Ltd.

SwapCols

Меняет местами столбцы в матрице.

```
bool matrix::SwapCols(
  const ulong  col1,     // индекс первого столбца
  const ulong  col2      // индекс второго столбца
   );
```

Параметры

col1

[in] Индекс первого столбца.

col2

[in] Индекс второго столбца.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
   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]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Split

Разделение матрицы на несколько подматриц.

```
bool matrix::Split(
  const ulong  parts,      // количество подматриц
  const int    axis,       // ось
  matrix&      splitted[]  // массив полученных подматриц
   );

void matrix::Split(
  const ulong& parts[],    // размеры подматриц
  const int    axis,       // ось
  matrix&      splitted[]  // массив полученных подматриц
   );
```

Параметры

parts

[in] Количество подматриц, на которое нужно разделить матрицу.

axis

[in] Ось. 0 - горизонтальная ось, 1 - вертикальная ось.

splitted

[out] Массив полученных подматриц.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Если указано количество подматриц, то получаются подматрицы одинакового размера. То есть, соответствующий размер матрицы (0 - количество строк, 1 - количество столбцов) должен делиться на число parts без остатка. Подматрицы разных размеров можно получить, используя массив размеров подматриц. Элементы массива размеров используются пока вся матрица не окажется поделённой. Если массив размеров закончился, а матрица ещё не до конца поделена, то неподелённый остаток будет последней подматрица.

Пример:

```
   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[]={2,2};
```

© 2000-2025, MetaQuotes Ltd.

```
 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]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Hsplit

Горизонтальное разделение матрицы на несколько подматриц. То же самое, что Split с axis=0

```
bool matrix::Hsplit(
  const ulong  parts,      // количество подматриц
  matrix&      splitted[]  // массив полученных подматриц
   );

void matrix::Hsplit(
  const ulong& parts[],    // размеры подматриц
  matrix&      splitted[]  // массив полученных подматриц
   );
```

Параметры

parts

[in] Количество подматриц, на которое нужно разделить матрицу.

splitted

[out] Массив полученных подматриц.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Если указано количество подматриц, то получаются подматрицы одинакового размера. То есть, количество строк должно делиться на число parts без остатка. Подматрицы разных размеров можно получить, используя массив размеров подматриц. Элементы массива размеров используются пока вся матрица не окажется поделённой. Если массив размеров закончился, а матрица ещё не до конца поделена, то неподелённый остаток будет последней подматрица.

Пример:

```
   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++)
```

© 2000-2025, 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]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Vsplit

Вертикальное разделение матрицы на несколько подматриц. То же самое, что Split с axis=1

```
bool matrix::Vsplit(
  const ulong  parts,      // количество подматриц
  matrix&      splitted[]  // массив полученных подматриц
   );

void matrix::Vsplit(
  const ulong& parts[],    // размеры подматриц
  matrix&      splitted[]  // массив полученных подматриц
   );
```

Параметры

parts

[in] Количество подматриц, на которое нужно разделить матрицу.

splitted

[out] Массив полученных подматриц.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Если указано количество подматриц, то получаются подматрицы одинакового размера. То есть, количество столбцов должно делиться на число parts без остатка. Подматрицы разных размеров можно получить, используя массив размеров подматриц. Элементы массива размеров используются пока вся матрица не окажется поделённой. Если массив размеров закончился, а матрица ещё не до конца поделена, то неподелённый остаток будет последней подматрица.

Пример:

```
   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]);
```

© 2000-2025, 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]]

*/
```

© 2000-2025, MetaQuotes Ltd.

ArgSort

Косвенная сортировка матрицы или вектора.

```
vector vector::Sort(
  func_name  compare_func=NULL,  // функция сравнения
  T          context             // параметр для пользовательской функции сортировки
   );

matrix matrix::Sort(
  func_name  compare_func=NULL   // функция сравнения
  T          context             // параметр для пользовательской функции сортировки
   );

matrix matrix::Sort(
  const int  axis,               // ось для сортировки
  func_name  compare_func=NULL   // функция сравнения
  T          context             // параметр для пользовательской функции сортировки
   );
```

Параметры

axis

[in] Ось, по которой производится сортировка: 0 — горизонтальная, 1 — вертикальная.

func_name

[in] Компаратор. Вы можете указать одно из значений перечисления ENUM_SORT_MODE или собственную функцию сравнения. Если функция не указана, используется сортировка по возрастанию.

Пользовательская функция сравнения может быть двух видов:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Здесь T — тип матрицы или вектора, а TContex — тип переменной context, переданной в качестве дополнительного параметра в метод Sort.

context

[in] Дополнительный необязательный параметр, который можно передать в пользовательскую функцию сортировки.

Возвращаемое значение

Вектор или матрица с индексами сортированных элементов. Например, результат [4,2,0,1,3] говорит, что в нулевой позиции упорядоченного вектора должен быть быть элемент с индексом 4, в первой — элемент с индексом 2 и т.д.

© 2000-2025, MetaQuotes Ltd.

Sort

Сортировка матрицы или вектора по месту.

```
void vector::Sort(
  func_name  compare_func=NULL,  // функция сравнения
  T          context             // параметр для пользовательской функции сортировки
   );

void matrix::Sort(
  func_name  compare_func=NULL   // функция сравнения
  T          context             // параметр для пользовательской функции сортировки
   );

void matrix::Sort(
  const int  axis,               // ось для сортировки
  func_name  compare_func=NULL   // функция сравнения
  T          context             // параметр для пользовательской функции сортировки
   );
```

Параметры

axis

[in] Ось, по которой производится сортировка: 0 — горизонтальная, 1 — вертикальная.

func_name

[in] Компаратор. Вы можете указать одно из значений перечисления ENUM_SORT_MODE или собственную функцию сравнения. Если функция не указана, используется сортировка по возрастанию.

Пользовательская функция сравнения может быть двух видов:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Здесь T — тип матрицы или вектора, а TContex — тип переменной context, переданной в качестве дополнительного параметра в метод Sort.

context

[in] Дополнительный необязательный параметр, который можно передать в пользовательскую функцию сортировки.

Возвращаемое значение

Нет. Сортировка производится по месту — применяется к данным матрицы/вектора, для которого вызван метод Sort.

Пример:

```
//+------------------------------------------------------------------+
//| Функция сортировки                                               |
//+------------------------------------------------------------------+
```

© 2000-2025, 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);
  }
//+------------------------------------------------------------------+
//| Функция запуска скрипта                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- заполнение вектора
   vector v(100);
   //--- сортировка по возрастанию
   v.Sort(MyDoubleComparator);   // здесь используется дополнительный параметр со знач
   Print(v);
   // сортировка по убыванию
   v.Sort(MyDoubleComparator,1); // здесь дополнительный параметр '1' явно указывается
   Print(v);
  }
```

© 2000-2025, MetaQuotes Ltd.

Математические операции над матрицами и векторами

Математические операции — сложение, вычитание, умножение и деление — можно производить над матрицами и векторами почленно.

Математические функции изначально были предназначены для проведения математических операций над скалярными величинами. Теперь большая часть этих функций может использоваться и с новыми типами данных — матрицами и векторами — MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh. В этом случае матрица или вектор обрабатываются почленно. Пример:

```
//---
  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]]
  */
```

В случае MathMod и MathPow в качестве второго параметра может быть использован как скаляр, так и матрица или вектор соответствующего размера.

Покажем на примере, как вычислить стандартное отклонение с помощью математических функций над вектором.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- используем инициализирующую функцию для заполнения вектора
  vector r(10, ArrayRandom); // массив случайных чисел от 0 до 1
//--- вычислим среднее значение
  double avr=r.Mean();       // среднее значение массива
  vector d=r-avr;            // вычислим массив отклонений от среднего значения
  Print("avr(r)=", avr);
  Print("r=", r);
  Print("d=", d);
  vector s2=MathPow(d, 2);   // массив квадратов отклонений
  double sum=s2.Sum();       // сумма квадратов отклонений
//--- вычислим стандартное отклонение 2-мя способами
  double std=MathSqrt(sum/r.Size());
  Print(" std(r)=", std);
```

© 2000-2025, 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
*/
//+------------------------------------------------------------------+
//| Заполняет вектор случайными значениями                           |
//+------------------------------------------------------------------+
void ArrayRandom(vector& v)
 {
  for(ulong i=0; i<v.Size(); i++)
    v[i]=double(MathRand())/32767.;
 }
```

© 2000-2025, MetaQuotes Ltd.

Математические операции

Математические операции — сложение, вычитание, умножение и деление — можно производить над матрицами и векторами почленно.

Обе матрицы или оба вектора должны быть одного и того же типа и иметь одинаковые размеры. Каждый член матрицы оперирует с соответствующим членом второй матрицы.

В качестве второго слагаемого (множителя, вычитаемого, делителя) можно также использовать скаляр соответствующего типа (double, float или complex). В этом случае каждый член матрицы или вектора будет оперировать с указанным скаляром.

```
  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;    // произведение Адамара (Hadamard product) не
  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;

//--- возможны операции по месту
  matrix_a+=matrix_b;
  matrix_a/=2;
```

Те же самые операции доступны и для векторов.

© 2000-2025, MetaQuotes Ltd.

Математические функции

Ряд математических функций: 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. В этом случае матрица или вектор обрабатываются почленно.

В случае MathMod и MathPow в качестве второго параметра может быть использован как скаляр, так и матрица или вектор соответствующего размера

```
   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++;
     }
```

© 2000-2025, MetaQuotes Ltd.

Произведения матриц и векторов

Матричные и векторные вычисления включают в себя:

- матричное умножение
- векторное умножение
- получение ковариационной матрицы
- вычисление кросс-корреляции двух векторов
- вычисление свертки двух векторов
- вычисление коэффициента корреляции

Функция	Действие
MatMul	Метод MatMul для выполнения произведения матриц и векторов имеет несколько перегрузок
GeMM	Общее матричное произведение двух матриц (General Matrix Multiply)
Power	Возводит квадратную матрицу в указанную целочисленную степень
Dot	Скалярное произведение двух векторов
Kron	Возвращает произведение Кронекера двух матриц, матрицы и вектора, вектора и матрицы или двух векторов
Inner	Внутреннее произведение двух матриц
Outer	Вычисляет внешнее произведение двух матриц или двух векторов
CorrCoef	Вычисляет коэффициент корреляции Пирсона (линейный коэффициент корреляции)
Cov	Вычисляет ковариационную матрицу
Correlate	Вычисляет значения взаимнокорреляционной функции (кросс-корреляцию) двух векторов
Convolve	Возвращает дискретную линейную свертку двух векторов последовательностей

© 2000-2025, MetaQuotes Ltd.

MatMul

Метод MatMul для выполнения произведения матриц и векторов имеет несколько перегрузок.

Умножение матрицы на матрицу: matrix[M][K] * matrix[K][N] = matrix[M][N]

```
matrix  matrix::MatMul(
  const matrix&  b      // вторая матрица
   );
```

Умножение вектора на матрицу: horizontal vector[K] * matrix[K][N] = horizontal vector[N]

```
vector  vector::MatMul(
  const matrix&  b      // матрица
   );
```

Умножение матрицы на вектор: matrix[M][K] * vertical vector[K] = vertical vector[M]

```
vector  matrix::MatMul(
  const vector&  b      // вектор
   );
```

Скалярное умножение векторов: horizontal vector * vertical vector = dot value

```
scalar  vector::MatMul(
  const vector&  b      // второй вектор
   );
```

Параметры

b

[in] Матрица или вектор.

Возвращаемое значение

Матрица, вектор или скаляр в зависимости от используемого метода.

Примечание

Перемножаемые матрицы должны быть совместимы, количество столбцов первой матрицы должно быть равно количеству строк второй матрицы. Перемножение матриц некоммутативно, умножение первой матрицы на вторую в общем случае не равно умножению второй матрицы на первую.

Произведение матриц состоит из всех возможных комбинаций скалярных произведений вектор-строк первой матрицы и вектор-столбцов второй матрицы.

При скалярном умножении векторов их длины должны совпадать.

При умножении вектора и матрицы длина вектора должна точно соответствовать количеству столбцов матрицы.

© 2000-2025, MetaQuotes Ltd.

Наивный алгоритм перемножения матриц на 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);
  }
```

Пример умножения матриц

```
   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]]
*/
```

Пример умножения горизонтального вектора на матрицу

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

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим матрицу 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   vector v3 = {1, 2, 3};
   Print("Произведение горизонтального вектора v на матрицу m[3,5]");
   Print("Слева вектор v3 = ", v3);
   Print("Справа матрица m35 = \n", m35);
   Print("v3.MatMul(m35) = горизонтальный вектор v[5] \n", v3.MatMul(m35));

   /* Результат
    Произведение горизонтального вектора v3 на матрицу m[3,5]
    Слева вектор v3 = [1,2,3]
    Справа матрица m35 =
    [[0,1,2,3,4]
     [5,6,7,8,9]
     [10,11,12,13,14]]
    v3.MatMul(m35) = горизонтальный вектор v[5]
    [40,46,52,58,64]
   */
  }
//+------------------------------------------------------------------+
//|  Заполнение матрицы нарастающими значениями                      |
//+------------------------------------------------------------------+
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;
        }
     }
//---
  }
```

Пример умножения матрицы на вертикальный вектор

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

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим матрицу 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   Print("Произведение матрицы m[3,5] на вертикальный вектор v[5]");
   vector v5 = {1,2,3,4,5};
   Print("Слева m35 = \n",m35);
   Print("Справа v5 = ",v5);
   Print("m35.MatMul(v5) = вертикальный вектор v[3] \n",m35.MatMul(v5));

   /* Результат
   Произведение матрицы m[3,5] на вертикальный вектор v[5]
   Слева m35 =
   [[0,1,2,3,4]
    [5,6,7,8,9]
    [10,11,12,13,14]]
   Справа v5 = [1,2,3,4,5]
   m35.MatMul(v5) = вертикальный вектор v[3]
   [40,115,190]
   */
  }
//+------------------------------------------------------------------+
//|  Заполнение матрицы нарастающими значениями                      |
//+------------------------------------------------------------------+
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;
        }
     }
//---
  }
```

Пример скалярного умножения векторов

```
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- скалярное умножение горизонтального и вертикального векторов
   vector a= {1, 2, 3};  // горизонтальный вектор
   vector b= {4, 5, 6};  // вертикальный вектор
   Print("a = ", a);
   Print("b = ", b);
   Print("1) a.MatMul(b) = ", a.MatMul(b));
   //--- покажем, что метод Dot дает такой же результат
   Print("2) a.Dot(b) = ", a.Dot(b));

   /* Результат
   a = [1,2,3]
   b = [4,5,6]
   1) a.MatMul(b) = 32.0
   2) a.Dot(b) = 32.0
    */
  }
```

Смотри также

Dot, GeMM

© 2000-2025, MetaQuotes Ltd.

GeMM

Метод GeMM реализует общее матричное произведение двух матриц (General Matrix Multiply). В общем виде выражение записывается как C ← α A B + β C , где матрицы A и B могут быть необязательно транспонированы. При обычном умножении матриц AB (MatMul) предполагается, что коэффициент *alpha* равен единице, а *beta* - нулю.

Главное отличие GeMM от MatMul с точки зрения эффективности, состоит в том, что MatMul всегда создаёт новый объект матрицы/вектора, в то время как GeMM работает с уже существующим объектом матрицы, который не пересоздаётся. Поэтому, если при использовании с GeMM заранее распределить память для соответствующей матрицы и работать потом с одними и теми же размерами матриц, то не будет никакого перераспределения памяти. Это может быть очень важным преимуществом в пользу GeMM при массовых вычислениях, например, при оптимизации в тестере стратегий или тренировке нейронной сети.

У GeMM также 4 перегрузки, как и у метода MatMul. Но при этом семантика 4-й перегрузки была изменена для того, чтобы можно было перемножать вертикальные и горизонтальные векторы.

В существующем объекте матрицы/вектора необязательно заранее распределять память под данные. При первом вызове GeMM память будет распределена и заполнена нулями.

Умножение матрицы на матрицу: `matrix C[M][N] = α * ( matrix A[M][K] * matrix B[K][N]) + β * matrix C[M][N]`

```
bool  matrix::GeMM(
  const matrix &A,     // первая матрица
  const matrix &B,     // вторая  матрица
  double alpha,        // множитель альфа для произведения матриц AB
  double beta,         // множитель бета для матрицы C
  uint   flags         // комбинация значений перечисления ENUM_GEMM (битовое "или"), к
   );
```

Умножение вектора на матрицу: `vector C[N] = α * ( vector A[K] * matrix B[K][N]) + β * vector C[N]`

```
bool  vector::GeMM(
  const vector &A,     // горизонтальный вектор
  const matrix &B,     // матрица
  double alpha,        // множитель альфа для произведения AB
  double beta,         // множитель бета для вектора C
  uint   flags         // значение перечисления ENUM_GEMM,  которое определяет транспон
   );
```

Умножение матрицы на вектор: `vector C[M] = α * ( matrix A[M][K] * vector B[K] * ) + β * vector C[M]`

```
bool  vector::GeMM(
  const matrix &A,     // матрица
  const vector &B,     // вертикальный вектор
  double alpha,        // множитель альфа для произведения AB
  double beta,         // множитель бета для вектора C
  uint   flags         // значение перечисления ENUM_GEMM,  которое определяет транспон
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Умножение вектора на вектор: `matrix C[M][N] = α * ( vector A[M] * vector B[N] * ) + β * matrix C[M][N].` Данная перегрузка возвращает матрицу, в отличии от метода MatMul, который возвращает скаляр.

```
bool  matrix::GeMM(
  const vector &A,     // первый вектор
  const vector &B,     // второй  вектор
  double alpha,        // множитель альфа для произведения AB
  double beta,         // множитель бета для матрицы C
  uint   flags         // значение перечисления ENUM_GEMM,  которое определяет транспон
   );
```

Параметры

A

[in] Матрица или вектор.

B

[in] Матрица или вектор.

alpha

[in] Множитель альфа для произведения AB.

beta

[in] Множитель бета для результирующей матрицы C.

flags

[in] Значение перечисления ENUM_GEMM, которое определяет транспонированность матриц A, B и C.

Возвращаемое значение

true в случае успешного выполнения операции, иначе false.

ENUM_GEMM

Перечисление флагов для метода GeMM.

Идентификатор	Описание
TRANSP_A	Использовать транспонированную матрицу A
TRANSP_B	Использовать транспонированную матрицу B
TRANSP_C	Использовать транспонированную матрицу C

Примечание

© 2000-2025, MetaQuotes Ltd.

В качестве параметров A и B могут использоваться матрицы и векторы типа float, double и complex. Таким образом, шаблонные варианты метода GeMM выглядят так:

```
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)
```

В общем случае функция общего матричного умножения описывается как:

```
C[m,n] = α *Sum(A[m,k]*B[k,n]) + β*C[m,n]
```

где матрица A имеет размер M x K, матрица B — K x N, и матрица C — M x N.

Таким образом, перемножаемые матрицы должны быть совместимы, количество столбцов первой матрицы должно быть равно количеству строк второй матрицы. Перемножение матриц некоммутативно, умножение первой матрицы на вторую в общем случае не равно умножению второй матрицы на первую.

Пример:

```
void OnStart()
  {
   vector vector_a= {1, 2, 3, 4, 5};
   vector vector_b= {4, 3, 2, 1};
   matrix matrix_c;
//--- вычислим GeMM для двух векторов
   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]]
   */
//--- создадим матрицы в виде векторов
   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:
```

© 2000-2025, MetaQuotes Ltd.

```
   [[1]
   [2]
   [3]
   [4]
   [5]]
   matrix_b:
   [[4,3,2,1]]
   */
//-- вычислим GeMM для двух матриц и получим такой же результат
   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]]
   */
  }
```

Смотри также

MatMul

© 2000-2025, MetaQuotes Ltd.

Power

Возводит квадратную матрицу в указанную целочисленную степень.

```
matrix matrix::Power(
  const int  power      // степень
   );
```

Параметры

power

[in] Показатель степени может быть любым целым числом: положительным, отрицательным или нулем.

Возвращаемое значение

Матрица.

Примечание

Полученная матрица имеет тот же самый размер, что и исходная матрица. В случае возведения матрицы в степень 0 возвращается единичная матрица. Положительная степень n означает, что исходная матрица умножается n раз на саму себя. Отрицательная степень -n означает, что исходная матрица сначала инвертируется, и потом инвертированная матрица n раз умножается на саму себя.

Простой алгоритм возведения матрицы в степень на MQL5:

```
bool MatrixPower(matrix& c, const matrix& a, const int power)
  {
//--- матрица должна быть квадратной
   if(a.Rows()!=a.Cols())
      return(false);
//--- размер результатной матрицы точно такой же
   ulong  rows=a.Rows();
   ulong  cols=a.Cols();
   matrix result(rows,cols);
//--- при нулевой степени возвращаем единичную матрицу
   if(power==0)
      result.Identity();
   else
     {
      //--- при отрицательной степени сначала инвертируем матрицу
      if(power<0)
        {
         matrix inverted=a.Inv();
         result=inverted;
         for(int i=-1; i>power; i--)
            result=result.MatMul(inverted);
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       else
         {
          result=a;
          for(int i=1; i<power; i++)
             result=result.MatMul(a);
         }
      }
//---
   c=result;
   return(true);
  }
```

Пример на 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]]
  */
```

Пример на Python:

```
import numpy as np
from numpy.linalg import matrix_power
```

© 2000-2025, MetaQuotes Ltd.

```
# matrix equiv. of the imaginary unit
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.]]
```

© 2000-2025, MetaQuotes Ltd.

Dot

Скалярное произведение двух векторов.

```
double vector::Dot(
  const vector&  b      // второй вектор
   );
```

Параметры

b

[in] Вектор.

Возвращаемое значение

Скаляр.

Примечание

Dot-продукт для двух матриц — это ничто иное, как матричное произведение matrix::MatMul().

Простой алгоритм скалярного произведения векторов на 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);
  }
```

Пример на 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);
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

Пример на Python:

```
import numpy as np

a = [1, 0, 0, 1]
b = [4, 1, 2, 2]
print(np.dot(a, b))

>>> 6
```

© 2000-2025, MetaQuotes Ltd.

Kron

Возвращает произведение Кронекера двух матриц, матрицы и вектора, вектора и матрицы или двух векторов.

```
matrix matrix::Kron(
  const matrix&  b      // вторая матрица
   );

matrix matrix::Kron(
  const vector&  b      // вектор
   );

matrix vector::Kron(
  const matrix&  b      // матрица
   );

matrix vector::Kron(
  const vector&  b      // второй вектор
   );
```

Параметры

b

[in] Вторая матрица.

Возвращаемое значение

Матрица.

Примечание

Произведение Кронекера также называют блочным перемножением матриц.

Простой алгоритм произведения Кронекера для двух матриц на 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++)
```

© 2000-2025, 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);
  }
```

Пример на 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]]
  */
```

Пример на 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]]
```

© 2000-2025, MetaQuotes Ltd.

Inner

Внутреннее произведение двух матриц.

```
matrix matrix::Inner(
  const matrix&  b      // вторая матрица
   );
```

Параметры

b

[in] Матрица.

Возвращаемое значение

Матрица.

Примечание

Внутренний продукт для двух векторов — это ничто иное, как скалярное произведение двух векторов vector::Dot().

Простой алгоритм внутреннего произведения двух матриц на MQL5:

```
bool MatrixInner(matrix& c, const matrix& a, const matrix& b)
  {
//--- количество столбцов должно быть одинаковым
   if(a.Cols()!=b.Cols())
      return(false);
//--- размер результатной матрицы зависит от количества векторов в каждой из матриц
   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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Пример на 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]]
  */
```

Пример на 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))
```

© 2000-2025, MetaQuotes Ltd.

Outer

Вычисляет внешнее произведение двух матриц или двух векторов.

```
matrix matrix::Outer(
  const matrix&  b      // вторая матрица
   );

matrix vector::Outer(
  const vector&  b      // второй вектор
   );
```

Параметры

b

[in] Матрица.

Возвращаемое значение

Матрица.

Примечание

Внешнее произведение, как и произведение Кронекера, также является блочным перемножением матриц (и векторов).

Простой алгоритм внешнего произведения двух матриц на MQL5:

```
matrix MatrixOuter(const matrix& matrix_a, const matrix& matrix_b)
  {
//--- размер результатной матрицы зависит от размеров матриц
   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];
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//---
   return(matrix_c);
  }
```

Пример на 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]]
   */
```

Пример на Python:

```
import numpy as np
```

© 2000-2025, 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]]
```

© 2000-2025, MetaQuotes Ltd.

CorrCoef

Вычисляет коэффициент корреляции Пирсона (линейный коэффициент корреляции).

```
matrix matrix::CorrCoef(
  const bool    rowvar=true  // вектора наблюдений расположены в строках, если false -
);

scalar vector::CorrCoef(
  const vector&  b           // второй вектор
);
```

Параметры

rowvar

[in] Флаг, определяющий расположение векторов в матрице для вычисления попарной корреляции — горизонтально или вертикально. По умолчанию *rowvar* равно true, это означает, что векторы находятся в строках матрицы. Если *rowvar* равно false, то корреляций считается между столюцами матрицы.

b

[in] Второй вектор.

Возвращаемое значение

Значение коэффициенты корреляции для вектора.

Квадратная матрица значений коэффициентов корреляцит между векторами матрицы для матрицы. Размер матрицы соответствует количеству векторов в матрице, который могут располагаться по строкам (*rowvar*=true) или по столбцам ((*rowvar*=false).

Примечание

Коэффициент корреляции находится в диапазоне [-1, 1].

Из-за округления числа с плавающей запятой результирующий массив может не быть эрмитовым, диагональные элементы могут не равняться 1, а элементы могут не удовлетворять неравенству abs(a) <= 1. Действительная и мнимая части обрезаются до интервала [-1, 1], чтобы это исправить, но это не всегда помогает в сложных случаях.

Простой алгоритм вычисления коэффициента корреляции двух векторов на 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);

   ulong  i;
```

© 2000-2025, MetaQuotes Ltd.

```
   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;
//--- вычисление
   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;
     }
//--- проверка
   if(xv==0 || yv==0)
      return(0);
//--- вернем результат
   return(s/(MathSqrt(xv)*MathSqrt(yv)));
  }
```

Пример на 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();
   Print("transposed matrix cols correlation\n",matrix_a2.CorrCoef(false));
   matrixf matrix_a3={{1.0f, 2.0f, 3.0f, 4.0f, 5.0f},
```

© 2000-2025, MetaQuotes Ltd.

```
                     {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]]
   */
```

Пример на 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))

transposed matrix cols correlation
[[1.         0.91499142]
```

© 2000-2025, MetaQuotes Ltd.

```
 [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.        ]]
```

© 2000-2025, MetaQuotes Ltd.

Cov

Вычисляет ковариационную матрицу.

```
matrix matrix::Cov(
  const bool     rowvar=true  // строки или столбцы векторов наблюдений
);

matrix matrix::Cov(
  const bool     rowvar,      // строки или столбцы векторов наблюдений
  const int      ddof         // сдвиг степени свободы
);

matrix vector::Cov(
  const vector&  b            // второй вектор
);

matrix vector::Cov(
  const vector&  b,           // второй вектор
  const int      ddof         // сдвиг степени свободы
);
```

Параметры

rowwar

[in] Если значение *rowwar* равно true (по умолчанию), то каждая строка представляет собой переменную, а наблюдения располагаются по столбцам. В противном случае (если false), строки содержат наблюдения, а столбцы — переменные (то есть матрица транспонируется).

b

[in] Второй вектор.

ddof

[in] "Delta Degrees of Freedom": параметр, определяющий, на сколько уменьшается число степеней свободы. Делитель в формуле будет равен N - ddof, где N — число наблюдений. По умолчанию ddof = 1.

Примечание

Вычисляет ковариационную матрицу.

Простой алгоритм вычисления ковариационной матрицы двух векторов на 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());
//--- проверки
   if(n<=1)
```

© 2000-2025, MetaQuotes Ltd.

```
      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);
//--- вычисление
   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++)
         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);
  }
```

Пример на 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));
```

© 2000-2025, MetaQuotes Ltd.

```
/*
 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,-4.389,8.7361999999999998]]
 covariation vectors
 [[2.144133333333333,-4.286]
  [-4.286,11.71]]
*/
```

Пример на Python:

```
import numpy as np
matrix_a=np.array([[3,-2.1],[1.1,-1],[0.12,4.3]])
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      ]]
```

© 2000-2025, MetaQuotes Ltd.

Correlate

Вычисляет значения взаимнокорреляционной функции (кросс-корреляцию) двух векторов.

```
vector vector::Correlate(
  const vector&           v,        // вектор
  ENUM_VECTOR_CONVOLVE    mode      // режим
   );
```

Параметры

v

[in] Второй вектор.

mode

[in] Параметр mode определяет режим вычисления линейной свертки. Значение из перечисления ENUM_VECTOR_CONVOLVE.

Возвращаемое значение

Кросс-корреляция двух векторов.

Примечание

Параметр mode определяет режим вычисления линейной свертки.

Простой алгоритм вычисления кросс-корреляции двух векторов на 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);
```

© 2000-2025, 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);
  }
```

Пример на 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
```

© 2000-2025, 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]
  */
```

Пример на 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]
```

© 2000-2025, MetaQuotes Ltd.

Convolve

Возвращает дискретную линейную свертку двух векторов последовательностей.

```
vector vector::Convolve(
  const vector&            v,         // вектор
  ENUM_VECTOR_CONVOLVE     mode       // режим
   );
```

Параметры

v

[out] Второй вектор.

mode

[in] Параметр mode определяет режим вычисления линейной свертки ENUM_VECTOR_CONVOLVE.

Возвращаемое значение

Дискретная линейная свертка двух векторов.

Простой алгоритм вычисления конволюции двух векторов на 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();
```

© 2000-2025, MetaQuotes Ltd.

```
   int    size=MathMax(m,n);
   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);
  }
```

Пример на 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));
```

© 2000-2025, MetaQuotes Ltd.

```
  Print("valid\n", a.Convolve(b, VECTOR_CONVOLVE_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]
  */
```

Пример на 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]
```

© 2000-2025, MetaQuotes Ltd.

Преобразования матриц

Разложение (декомпозиция) матриц — задача, которая может возникать в следующих случаях:

- как промежуточный этап в процессе решения систем линейных уравнений
- для обращения матриц
- для вычисления определителей
- при отыскании собственных значений и собственных векторов матрицы
- для вычисления аналитических функций от матриц
- при использовании метода наименьших квадратов
- при численном решении дифференциальных уравнений

При этом, в зависимости от решаемой задачи, используются различные виды разложения (декомпозиции) матриц.

Функция	Действие
Cholesky	Вычисляет декомпозицию Холецкого
Eig	Вычисляет собственные значения и правые собственные векторы квадратной матрицы
EigVals	Вычисляет собственные значения общей матрицы
LU	Выполняет LU-факторизацию матрицы как произведения нижнетреугольной матрицы и верхнетреугольной матрицы
LUP	Выполняет LUP-факторизацию с частичным поворотом, которая является LU-факторизацией только с перестановками строк: PA=LU
QR	Вычисляет qr-факторизацию матрицы
SVD	Вычисляет разложение по сингулярным значениям

© 2000-2025, MetaQuotes Ltd.

Cholesky

Вычисляет декомпозицию Холецкого.

```
bool matrix::Cholesky(
   matrix&  L      // matrix
   );
```

Параметры

L

[out] Нижнетреугольная матрица.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Возвращает разложение Холецкого, L * L.H, квадратной матрицы a, где L является нижнетреугольной матрицей, а .H является эрмитово-сопряженной матрицей (который является обычным транспонированием при вещественной a). a должна быть эрмитовой (симметричной, если она вещественная) и положительно определенной. Функция не проверяет, является ли a эрмитовой или нет. Кроме того, используются только нижнетреугольные и диагональные элементы a. Фактически возвращается только L.

Пример:

```
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]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Eig

Вычисляет собственные значения и правые собственные векторы квадратной матрицы.

```
bool matrix::Eig(
  matrix&  eigen_vectors,          // матрица собственных векторов
  vector&  eigen_values            // вектор собственных значений
   );
```

Комплексное решение собственных значений и собственных векторов

```
bool matrix::Eig(
  matrix<complex>& eigen_vectors, // матрица собственных векторов
  vector<complex>& eigen_values   // вектор собственных значений
   );
```

Параметры

eigen_vectors

[out] Матрица вертикальных собственных векторов.

eigen_values

[out] Вектор собственных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Если при вычислении собственных значений встречается комплексное решение, то расчёт останавливается, при этом код ошибки устанавливается в 4019 (ERR_MATH_OVERFLOW). Чтобы получить полное решение в комплексном пространстве, необходимо использовать комплексную перегрузку метода Eig

Если комплексное собственное число имеет мнимую часть равную нулю, то это уже действительное собственное число. И это можно увидеть в примере ниже.

Пример:

```
void OnStart()
  {
   matrix matrix_a =
     {
        {-3.474589, 1.106384, -9.091977,-3.925227 },
        {-5.522139, 2.366887,-15.162351,-6.357512 },
        { 8.394926,-2.960067, 22.292115, 9.524129 },
        { 7.803242,-2.080287, 19.217706, 8.186645 }
     };

   matrix eigen_vectors;
```

© 2000-2025, MetaQuotes Ltd.

```
   vector eigen_values;

   bool res=matrix_a.Eig(eigen_vectors, eigen_values);
   Print("res=",res,"  error=",GetLastError());
   Print("Eigen vectors:\n",eigen_vectors, "\nEigen Values:\n",eigen_values);

//--- check correctness A * v = lambda * v
   int vectors=0;
   for(ulong n=0; n<eigen_values.Size(); n++)
     {
      vector eigen_vector=eigen_vectors.Col(n);
      vector vector_d1   =eigen_vector*eigen_values[n];
      vector vector_d2   =matrix_a.MatMul(eigen_vector);

      ulong errors=vector_d1.Compare(vector_d2,1e-13);
      if(errors==0)
         vectors++;
     }
   Print("vectors=",vectors);

//--- complex solution
   matrix<complex> eigen_vectors_c;
   vector<complex> eigen_values_c;
   ResetLastError();
   res=matrix_a.Eig(eigen_vectors_c,eigen_values_c);
   Print("res=",res,"  error=",GetLastError());
   Print("Eigen vectors:\n",eigen_vectors_c, "\nEigen Values:\n",eigen_values_c);

//--- check correctness A * v = lambda * v
   matrixc matrix_c;
   matrix_c.Assign(matrix_a);
   vectors=0;
   for(ulong n=0; n<eigen_values_c.Size(); n++)
     {
      vectorc eigen_vector_c=eigen_vectors_c.Col(n);
      vectorc vector_c1     =eigen_vector_c*eigen_values_c[n];
      vectorc vector_c2     =matrix_c.MatMul(eigen_vector_c);

      ulong errors=vector_c1.Compare(vector_c2,1e-13);
      if(errors==0)
         vectors++;
     }
   Print("vectors=",vectors);
  }
/* Результат
   res=true  error=4019
   Eigen vectors:
   [[0.2649667608713664]
    [0.4488818803991876]
```

© 2000-2025, MetaQuotes Ltd.

```
    [-0.6527335897527492]
    [-0.5497604331807768]]
   Eigen Values:
   [28.94158645962942]
   vectors=1
   res=true  error=0
   Eigen vectors:
   [[(0.2649667608713664,0),(0.2227392219204762,0.3745470492013296),(0.403285439089771
    [(0.4488818803991876,0),(-0.2613452530342438,-0.3685707727819327),(-0.396850612637
    [(-0.6527335897527492,0),(-0.3418479634708521,-0.3299830378162041),(-0.355302103111
    [(-0.5497604331807768,0),(0.523227993452828,0.3262508584080381),(0.351283559614366
   Eigen Values:
   [(28.94158645962942,0),(0.01022501104810897,0.02954980822331488),(0.010225011048108
   vectors=3
*/
```

© 2000-2025, MetaQuotes Ltd.

EigVals

Вычисляет собственные значения общей матрицы.

```
bool matrix::EigVals(
  vector&  eigen_values      // вектор собственных значений
   );
```

Параметры

eigen_values

[out] Вектор правых собственных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Единственная разница между EigVals и Eig — EigVals не рассчитывает собственные векторы, а только собственные значения.

© 2000-2025, MetaQuotes Ltd.

LU

Выполняет LU-факторизацию матрицы как произведения нижнетреугольной матрицы и верхнетреугольной матрицы.

```
bool matrix::LU(
  matrix&  L,      // нижнетреугольная матрица
  matrix&  U       // верхнетреугольная матрица
   );
```

Параметры

L

[out] Нижнетреугольная матрица.

U

[out] Верхнетреугольная матрица.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u;
//--- LU декомпозиция
   matrix_a.LU(matrix_l,matrix_u);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
//--- проверим правильность 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]
```

© 2000-2025, MetaQuotes Ltd.

```
    [5,2,6,7]
    [8,9,3,10]
    [11,12,14,4]]
  */
```

© 2000-2025, MetaQuotes Ltd.

LUP

Выполняет LUP-факторизацию с частичным поворотом, которая является LU-факторизацией только с перестановками строк: PA=LU.

```
bool  LUP(
  matrix&  L,      // нижнетреугольная матрица
  matrix&  U,      // верхнетреугольная матрица
  matrix&  P       // матрица перестановок
   );
```

Параметры

L

[out] Нижнетреугольная матрица.

U

[out] Верхнетреугольная матрица.

P

[out] Матрица перестановок.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
   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;
//--- LUP декомпозиция
   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);
//--- проверим правильность 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]
```

© 2000-2025, 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]]
  */
```

© 2000-2025, MetaQuotes Ltd.

QR

Вычисляет qr-факторизацию матрицы.

```
bool  QR(
  matrix&  Q,      // матрица с ортонормированными столбцами
  matrix&  R       // верхнетреугольная матрица
   );
```

Параметры

Q

[out] Матрица с ортонормированными столбцами. При включении режима complete результатом будет ортогональная/унитарная матрица в зависимости от того, является ли a вещественной/комплексной. В этом случае определитель может быть +/- 1. Если количество измерений во входном массиве больше 2, возвращается стек матриц с указанными выше свойствами.

R

[out] Верхнетреугольная матрица.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
//---  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]
```

© 2000-2025, MetaQuotes Ltd.

```
[-0.5374574785652648,-0.03745747856526496,0.7041241452319316,-0.4625425214347352]
[-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]]
*/
```

© 2000-2025, MetaQuotes Ltd.

SVD

Вычисляет разложение по сингулярным значениям.

```
bool matrix::SVD(
  matrix&  U,                    // унитарная матрица
  matrix&  V,                    // унитарная матрица
  vector&  singular_values       // вектор сингулярных значений
   );
```

Параметры

U

[out] Унитарная матрица порядка m, состоящая из левых сингулярных векторов.

V

[out] Унитарная матрица порядка n, состоящая из правых сингулярных векторов.

singular_values

[out] Сингулярные значения

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
  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);
//--- сделаем SVD-разложение
  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);

// блок проверки
//--- 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);
```

© 2000-2025, MetaQuotes Ltd.

```
  ulong errors=(int)b.Compare(matrix_usvt, 1e-9);
  double res=(errors==0);
  Print("errors=", errors);

//---- еще проверка
  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-17,0.9999999999999999]]
```

© 2000-2025, 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]
  */
 }
```

© 2000-2025, MetaQuotes Ltd.

Статистические методы

Методы для получения описательной статистики матриц и векторов.

Функция	Действие
ArgMax	Возвращает индекс максимального значения
ArgMin	Возвращает индекс минимального значения
Max	Возвращает максимальное значение в матрице/векторе
Min	Возвращает минимальное значение в матрице/векторе
Ptp	Возвращает диапазон значений матрицы/вектора или указанной оси матрицы
Sum	Выполняет суммирование элементов матрицы/вектора, которое так же может выполняться по указанной оси (осям)
Prod	Вычисляет произведение элементов матрицы/вектора, которое так же может выполняться по указанной оси
CumSum	Возвращает кумулятивную (накапливаемую) сумму элементов матрицы/вектора, в том числе и по заданной оси
CumProd	Возвращает кумулятивное (накапливаемое) произведение элементов массива/вектора, в том числе и по заданной оси
Percentile	Вычисляет указанный процентиль (перцентиль) значений элементов матрицы/вектора или элементов вдоль указанной оси
Quantile	Вычисляет указанный квантиль значений элементов матрицы/вектора или элементов вдоль указанной оси
Median	Вычисляет медиану элементов массива/вектора
Mean	Вычисляет среднее арифметическое значений элементов
Average	Вычисляет взвешенное среднее арифметическое значений матрицы/вектора
Std	Вычисляет среднеквадратичное (стандартное) отклонение значений элементов матрицы/вектора или указанной оси
Var	Вычисляет дисперсию значений элементов матрицы/вектора
LinearRegression	Вычисляет вектор/матрицу с рассчитанными значениями линейной регрессии

© 2000-2025, MetaQuotes Ltd.

ArgMax

Возвращает индекс максимального значения.

```
ulong vector::ArgMax();

ulong matrix::ArgMax();

vector matrix::ArgMax(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Индекс максимального значения.

Пример:

```
   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
   */
```

© 2000-2025, MetaQuotes Ltd.

ArgMin

Возвращает индекс минимального значения.

```
ulong vector::ArgMin();

ulong matrix::ArgMin();

vector matrix::ArgMin(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Индекс минимального значения.

Пример:

```
   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
   */
```

© 2000-2025, MetaQuotes Ltd.

Max

Возвращает максимальное значение в матрице/векторе.

```
double vector::Max();

double matrix::Max();

vector matrix::Max(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Максимальное значение в матрице/векторе.

Пример:

```
   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
   */
```

© 2000-2025, MetaQuotes Ltd.

Min

Возвращает минимальное значение в матрице/векторе.

```
double vector::Min();

double matrix::Min();

vector matrix::Min(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Минимальное значение в матрице/векторе.

Пример:

```
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
*/
```

© 2000-2025, MetaQuotes Ltd.

Ptp

Возвращает диапазон значений матрицы/вектора или указанной оси матрицы и эквивалентна результату Max() - Min(). Ptp - Peak to peak, от пика до пика.

```
double vector::Ptp();

double matrix::Ptp();

vector matrix::Ptp(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Вектор с диапазонами значений (максимум - минимум).

Пример:

```
   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
   */
```

© 2000-2025, MetaQuotes Ltd.

Sum

Выполняет суммирование элементов матрицы/вектора, которое так же может выполняться по указанной оси (осям).

```
double vector::Sum();

double matrix::Sum();

vector matrix::Sum(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Сумма элементов матрицы/вектора, которое так же может выполняться по указанной оси (осям).

Пример:

```
   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
   */
```

© 2000-2025, MetaQuotes Ltd.

Prod

Вычисляет произведение элементов матрицы/вектора, которое так же может выполняться по указанной оси.

```
double vector::Prod(
  const double  initial=1      // начальный множитель
   );

double matrix::Prod(
  const double  initial=1      // начальный множитель
   );

vector matrix::Prod(
  const int     axis,          // ось
  const double  initial=1      // начальный множитель
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

initial=1

[in] Начальный множитель.

Пример:

```
   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=[42,132,86.400000000000001]
```

© 2000-2025, MetaQuotes Ltd.

```
   rows_prod=[60,96,120,693]
   prod value 479001600.0
  */
```

© 2000-2025, MetaQuotes Ltd.

CumSum

Возвращает кумулятивную (накапливаемую) сумму элементов матрицы/вектора, в том числе и по заданной оси.

```
vector vector::CumSum();

vector matrix::CumSum();

matrix matrix::CumSum(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Накопленная сумма элементов по заданной оси.

Пример:

```
   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]
```

© 2000-2025, MetaQuotes Ltd.

```
    [6,11,15]
    [7,18,27]]
   cumsum values  [10,13,15,16,24,36,42,47,51,58,69,78]
   */
```

© 2000-2025, MetaQuotes Ltd.

CumProd

Возвращает кумулятивное (накапливаемое) произведение элементов массива/вектора, в том числе и по заданной оси.

```
vector vector::CumProd();

vector matrix::CumProd();

matrix matrix::CumProd(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось для каждого столбца (т.е. по строкам), 1 — вертикальная ось для каждой строки (т.е. по столбцам).

Возвращаемое значение

Совокупное произведение элементов по заданной оси.

Пример:

```
   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]
```

© 2000-2025, MetaQuotes Ltd.

```
    [1,8,96]
    [6,30,120]
    [7,77,693]]
   cumprod values  [10,30,60,60,480,5760,34560,172800,691200,4838400,53222400,47900160
   */
```

© 2000-2025, MetaQuotes Ltd.

Percentile

Вычисляет указанный процентиль (перцентиль) значений элементов матрицы/вектора или элементов вдоль указанной оси.

```
double vector::Percentile(
  const int  percent      //
   );

double matrix::Percentile(
  const int  percent      //
   );

vector matrix::Percentile(
  const int  percent,     //
  const int  axis         // ось
   );
```

Параметры

percent

[in] Вычисляемый процентиль, должен быть в диапазоне от 0 до 100 включительно.

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Процентиль: скалярный или векторный.

Примечание

Допустимые значения параметра percent находятся в диапазоне [0, 100]. Для расчёта перцентилей используется линейный алгоритм. Для правильного расчёта перцентилей последовательность должна быть отсортирована.

Пример:

```
   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);
```

© 2000-2025, 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
   */
```

© 2000-2025, MetaQuotes Ltd.

Quantile

Вычисляет указанный квантиль значений элементов матрицы/вектора или элементов вдоль указанной оси.

```
double vector::Quantile(
  const double quantile      // квантиль
   );

double matrix::Quantile(
  const double quantile      // квантиль
   );

vector matrix::Quantile(
  const double quantile,     // квантиль
  const int    axis          // ось
   );
```

Параметры

quantile

[in] Вычисляемый квантиль, должен быть в диапазоне от 0 до 100 включительно.

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось

Возвращаемое значение

Квантиль: скаляр или вектор.

Примечание

Диапазон значений параметра quantile принимает значения в интервале [0, 1] Для расчета квантилей используется линейный алгоритм. Для правильного расчета квантилей последовательность должна быть отсортирована.

Пример:

```
   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);

   /*
```

© 2000-2025, 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
   */
```

© 2000-2025, MetaQuotes Ltd.

Median

Вычисляет медиану элементов массива/вектора.

```
double vector::Median();

double matrix::Median();

vector matrix::Median(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Медиана: скалярная или векторная.

Примечание

Медиана — это такое значение, что ровно половина элементов матрицы/вектора окажется меньше него, а другая больше. То же самое, что Quantile(0.5) и Percentile(50). Для правильного расчёта перцентилей последовательность должна быть отсортирована.

Пример:

```
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]
```

© 2000-2025, MetaQuotes Ltd.

```
   rows_median [2,5,8,11]
   median value  6.5
   */
```

© 2000-2025, MetaQuotes Ltd.

Mean

Вычисляет среднее арифметическое значений элементов.

```
double vector::Mean();

double matrix::Mean();

vector matrix::Mean(
  const int  axis      // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Среднее арифметическое значений элементов.

Пример:

```
   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
   */
```

© 2000-2025, MetaQuotes Ltd.

Average

Вычисляет взвешенное среднее арифметическое значений матрицы/вектора.

```
double vector::Average(
  const vector& weigts      // вектор весов
   );

double matrix::Average(
  const matrix& weigts      // матрица весов
);

vector matrix::Average(
  const matrix& weigts,     // матрица весов
  const int     axis        // ось
   );
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Среднее арифметическое: скалярное или векторное.

Примечание

Матрица/вектор весов ассоциируется с основной матрицей/вектором.

Пример:

```
   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]
```

© 2000-2025, 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
```

© 2000-2025, MetaQuotes Ltd.

Std

Вычисляет среднеквадратичное (стандартное) отклонение значений элементов матрицы/вектора или указанной оси.

```
double vector::Std();

double vector::Std(
  const int  ddof      // сдвиг степени свободы
);

double matrix::Std();

vector matrix::Std(
  const int  axis      // ось
   );

vector matrix::Std(
  const int  axis,     // ось
  const int  ddof      // сдвиг степени свободы
);
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

ddof

[in] "Delta Degrees of Freedom": параметр, определяющий, на сколько уменьшается число степеней свободы. Делитель в формуле будет равен N - ddof, где N — число наблюдений. По умолчанию ddof = 0.

Возвращаемое значение

Стандартное отклонение: скалярное или векторное.

Примечание

Стандартное отклонение представляет собой корень квадратный из среднего значения квадратов отклонений от среднего, т. е. std = sqrt(mean(x)), где x = abs(a - a.mean())**2.

Среднеквадратичное отклонение обычно рассчитывается как x.sum() / N, где N = len(x).

Выражение с *ddof*=0 иногда называют "дисперсией генеральной совокупности". Если *ddof* > 0 (чаще всего используется значение 1), полученная величина называется "выборочной дисперсией".

Пример:

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
```

© 2000-2025, MetaQuotes Ltd.

```
   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]]
   cols_std [3.2403703,3.0310888,3.9607449]
   rows_std [3.5590262,4.5460606,0.81649661,1.6329932]
   std value  3.452052593231201
   */
```

© 2000-2025, MetaQuotes Ltd.

Var

Вычисляет дисперсию значений элементов матрицы/вектора.

```
double vector::Var();

double vector::Std(
  const int  ddof      // сдвиг степени свободы
);

double matrix::Var();

vector matrix::Var(
  const int  axis      // ось
   );

vector matrix::Std(
  const int  axis,     // ось
  const int  ddof      // сдвиг степени свободы
);
```

Параметры

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

ddof

[in] “Delta Degrees of Freedom”: параметр, определяющий, на сколько уменьшается число степеней свободы. Делитель в формуле будет равен N - ddof, где N — число наблюдений. По умолчанию ddof = 0.

Возвращаемое значение

Дисперсия: скалярная или векторная.

Примечание

Дисперсия представляет собой среднеквадратичное отклонение от среднего, т.е. var = mean(x), где x = abs(a - a.mean())**2.

Среднее значение обычно рассчитывается как x.sum() / N, где N = len(x).

Выражение с *ddof*=0 иногда называют "дисперсией генеральной совокупности". Если *ddof* > 0 (чаще всего используется значение 1), полученная величина называется "выборочной дисперсией".

Пример:

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);
```

© 2000-2025, MetaQuotes Ltd.

```
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]
var value  11.916666984558105
*/
```

© 2000-2025, MetaQuotes Ltd.

LinearRegression

Вычисляет вектор/матрицу с рассчитанными значениями линейной регрессии.

```
vector vector::LinearRegression();

matrix matrix::LinearRegression(
  ENUM_MATRIX_AXIS  axis=AXIS_NONE      // ось, вдоль которой считаетя регрессия
   );
```

Параметры

axis

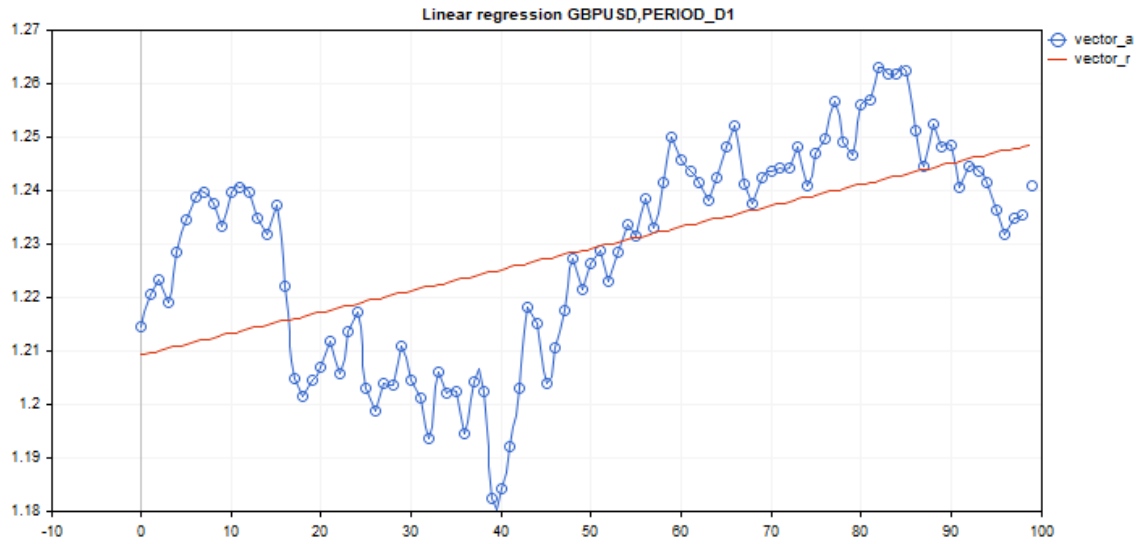
[in] Указание оси, вдоль которой считается регрессия. Значение из перечисления ENUM_MATRIX_AXIS (AXIS_HORZ — горизонтальная ось, AXIS_VERT — вертикальная ось).

Возвращаемое значение

Вектор или матрица с рассчитанными значениями линейной регрессии.

Примечание

Для вычисления линейной регрессии используется стандартное уравнение регрессии: y (x) = a * x + b, где a — это наклон линии, а b — ее сдвиг по оси Y.



Пример:

```
#include <Graphics\Graphic.mqh>

#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

//+------------------------------------------------------------------+
```

© 2000-2025, 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);
```

© 2000-2025, 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

Перечисление для указания оси во всех статистических функциях для матриц.

Идентификатор	Описание
AXIS_NONE	Ось не задана, вычисление производится по всем элементам матрицы, как если бы это был вектор (смотри метод Flat).
AXIS_HORZ	Горизонтальная ось
AXIS_VERT	Вертикальная ось

© 2000-2025, MetaQuotes Ltd.

Методы характеристик

Методы для получения следующих характеристик матрицы:

- количество строк
- количество столбцов
- норма
- число обусловленности
- детерминант
- ранг матрицы
- след
- спектр

Функция	Действие
Rows	Возвращает количество строк в матрице
Cols	Возвращает количество столбцов в матрице
Size	Возвращает размер вектора
Norm	Возвращает норму матрицы или вектора
Cond	Вычисляет число обусловленности матрицы
Det	Вычисляет определитель квадратной невырожденной матрицы
SLogDet	Вычисляет знак и логарифм определителя матрицы
Rank	Возвращает ранг матрицы, используя метод Гаусса
Trace	Возвращает сумму элементов по диагоналям матрицы
Spectrum	Возвращает спектр матрицы как набор ее собственных значений из произведения AT*A

© 2000-2025, MetaQuotes Ltd.

Rows

Возвращает количество строк в матрице.

```
ulong matrix::Rows()
```

Возвращаемое значение

Значение типа integer.

Пример:

```
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
*/
```

© 2000-2025, MetaQuotes Ltd.

Cols

Возвращает количество столбцов в матрице.

```
ulong matrix::Cols()
```

Возвращаемое значение

Значение типа integer.

Пример:

```
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
*/
```

© 2000-2025, MetaQuotes Ltd.

Size

Возвращает размер вектора.

```
ulong vector::Size()
```

Возвращаемое значение

Значение типа integer.

Пример:

```
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]
*/
```

© 2000-2025, MetaQuotes Ltd.

Norm

Возвращает норму матрицы или вектора.

```
double vector::Norm(
  const ENUM_VECTOR_NORM  norm,     // норма вектора
  const int               norm_p=2  // число p-norm в случае VECTOR_NORM_P
   );

double matrix::Norm(
  const ENUM_MATRIX_NORM  norm      // норма матрицы
   );
```

Параметры

norm

[in] Порядок нормы

Возвращаемое значение

Норма матрицы или вектора.

Примечание

- VECTOR_NORM_INF — максимальное абсолютное значение среди элементов вектора.
- VECTOR_NORM_MINUS_INF — минимальное абсолютное значение вектора.
- VECTOR_NORM_P — P-норма вектора. Если norm_p=0, то это количество ненулевых элементов вектора; norm_p=1 - сумма абсолютных значений элементов вектора; norm_p=2 - квадратный корень из суммы квадратов значений элементов вектора. Значение параметра norm_p может быть отрицательным.
- MATRIX_NORM_FROBENIUS — квадратный корень из суммы квадратов значений элементов матрицы. Норма Фробениуса и векторная P2-норма являются согласованными.
- MATRIX_NORM_SPECTRAL — максимальное значение спектра матрицы.
- MATRIX_NORM_NUCLEAR — сумма сингулярных значений матрицы.
- MATRIX_NORM_INF — максимальная векторная p1-норма среди вертикальных векторов матрицы. Матричная inf-норма и векторная inf-норма являются согласованными.
- MATRIX_NORM_MINUS_INF — минимальная векторная p1-норма среди вертикальных векторов матрицы.
- MATRIX_NORM_P1 — максимальная векторная p1-норма среди горизонтальных векторов матрицы.
- MATRIX_NORM_MINUS_P1 — минимальная векторная p1-норма среди горизонтальных векторов матрицы.
- MATRIX_NORM_P2 — наибольшее сингулярное значение матрицы.
- MATRIX_NORM_MINUS_P2 — наименьшее сингулярное значение матрицы.

Простой алгоритм расчёта P-нормы вектора на MQL5:

```
double VectorNormP(const vector& v,int norm_value)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   ulong  i;
   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);
  }
```

Пример на 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]]
```

© 2000-2025, MetaQuotes Ltd.

```
  matrix b
  [[-4,-3,-2]
  [-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
  */
```

Пример на 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
```

© 2000-2025, MetaQuotes Ltd.

Cond

Вычисляет число обусловленности матрицы.

```
double matrix::Cond(
  const ENUM_MATRIX_NORM  norm      // норма матрицы
   );
```

Параметры

norm

[in] Порядок нормы из перечисления ENUM_MATRIX_NORM.

Возвращаемое значение

Число обусловленности матрицы. Может быть бесконечным.

Примечание

Число обусловленности x определяется как произведение нормы x и ее обратной x [1]. При этом норма может быть обычной L2-нормой (корень из суммы квадратов) или одной из ряда других матричных норм.

Числом (или мерой) обусловленности матрицы называют величину K, равную произведению норм матрицы A и ее обратной. Матрицы с большим числом обусловленности называются плохо обусловленными и, наоборот, матрицы с малым значением числа называются хорошо обусловленными. Обратная матрица получается при помощи псевдоинверсии, чтобы не ограничиваться условием квадратности и невырожденности матрицы.

Исключением является спектральное число обусловленности.

Простой алгоритм расчёта спектрального числа обусловленности на 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;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

Пример на 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
*/
```

Пример на 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))
```

© 2000-2025, 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
```

© 2000-2025, MetaQuotes Ltd.

Det

Вычисляет определитель квадратной невырожденной матрицы.

```
double matrix::Det()
```

Возвращаемое значение

Определитель (детерминант) матрицы.

Примечание

Определители матриц 2 и 3 порядков считаются по правилу Саррюса. d2=a11*a22-a12*a21; d3=a11*a22*a33+a12*a23*a31+a13*a21*a32-a13*a22*a31-a11*a23*a32-a12*a21*a33

Определитель считается методом Гаусса путём приведения матрицы к верхнетреугольному виду. Определитель верхнетреугольной матрицы равен произведению членов главной диагонали.

Если хотя бы одна строка (столбец) матрицы нулевая, то определитель равен нулю.

Если две (или несколько) строки (столбца) матрицы линейно зависимы, то её определитель равен нулю.

Определитель матрицы равен произведению её собственных значений.

Пример на 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
*/
```

Пример на 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
 [[1 2]
```

© 2000-2025, MetaQuotes Ltd.

```
 [3 4]]

np.linalg.det(a)
 -2.0000000000000004
```

© 2000-2025, MetaQuotes Ltd.

SLogDet

Вычисляет знак и логарифм определителя матрицы.

```
double matrix::SLogDet(
  int&  sign      // знак
   );
```

Параметры

sign

[out] Знак определителя (детерминанта). Если значение sign чётное, то детерминант положительный.

Возвращаемое значение

Число, представляющее знак определителя.

Примечание

Определитель считается методом Гаусса путём приведения матрицы к верхнетреугольному виду. Определитель верхнетреугольной матрицы равен произведению членов главной диагонали. Логарифм произведения равен сумме логарифмов. Поэтому в случае переполнения при вычислении детерминанта можно воспользоваться методом SLogDet.

Если значение sign чётное, то детерминант положительный.

Пример:

```
a = np.array([[1, 2], [3, 4]])  (sign, logdet) = np.linalg.slogdet(a)  (sign, logdet)
```

© 2000-2025, MetaQuotes Ltd.

Rank

Возвращает ранг матрицы, используя метод Гаусса.

```
int  Rank()
```

Возвращаемое значение

Ранг матрицы.

Примечание

Рангом системы строк (столбцов) матрицы A с m строками и n столбцами называется максимальное число линейно независимых строк (столбцов). Несколько строк (столбцов) называются линейно независимыми, если ни одна из них не выражается линейно через другие. Ранг системы строк всегда равен рангу системы столбцов, и это число называется рангом матрицы.

Пример на 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.;     // дефицит матрицы
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 размерность - ранг 1, если только не все "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]
```

© 2000-2025, 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
  */
```

Пример на 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.]]
```

© 2000-2025, MetaQuotes Ltd.

```
matrix_rank(I)= 3

b
 [1. 1. 1. 1.]
matrix_rank(b)= 1

zeroes
 [0. 0. 0. 0.]
matrix_rank(zeros)= 0
```

© 2000-2025, MetaQuotes Ltd.

Trace

Возвращает сумму элементов по диагоналям матрицы.

```
double matrix::Trace()
```

Возвращаемое значение

Сумма элементов по диагонали.

Примечание

След матрицы равен сумме её собственных значений.

Пример на 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
  */
```

Пример на 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
```

© 2000-2025, MetaQuotes Ltd.

Spectrum

Возвращает спектр матрицы как набор ее собственных значений из произведения AT*A.

```
vector matrix::Spectrum()
```

Возвращаемое значение

Спектр матрицы как вектор собственных значений матрицы.

Пример:

```
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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Матричные методы решения систем линейных уравнений

Методы для решения систем линейных уравнений и вычисления обратной матрицы.

Функция	Действие
Solve	Решает линейное матричное уравнение или систему линейных алгебраических уравнений
LstSq	Решает систему линейных алгебраических уравнений приблизительно (для неквадратных или вырожденных матриц)
Inv	Вычисляет мультипликативную обратную матрицу по отношению к квадратной невырожденной матрице методом Жордана-Гаусса
PInv	Вычисляет псевдообратную матрицу методом Мура-Пенроуза

© 2000-2025, MetaQuotes Ltd.

Solve

Решает линейное матричное уравнение или систему линейных алгебраических уравнений.

```
vector matrix::Solve(
  const vector  b      // значения ординаты или "зависимой переменной"
   );
```

Параметры

b

[in] Значение ординаты или "зависимая переменная". (Вектор свободных членов).

Возвращаемое значение

Вектор с решением системы a * x = b.

Примечание

Если хотя бы одна строка (столбец) матрицы нулевая, то система не имеет решения.

Если две (или несколько) строки (столбца) матрицы линейно зависимы, то система не имеет решения.

Пример:

```
//--- решение СЛАУ
   vector_x=matrix_a.Solve(vector_b);
//--- проверим правильность a * x = b
   result_vector=matrix_a.MatMul(vector_x);
   errors=vector_b.Compare(result_vector,1e-12);
```

© 2000-2025, MetaQuotes Ltd.

LstSq

Решает систему линейных алгебраических уравнений приблизительно (для неквадратных или вырожденных матриц).

```
vector matrix::LstSq(
  const vector  b      // значения ординаты или "зависимой переменной"
   );
```

Параметры

b

[in] Значение ординаты или "зависимая переменная". (Вектор свободных членов)

Возвращаемое значение

Вектор с решением системы a * x = b. Это справедливо только для систем, имеющих точное решение.

Пример:

```
   matrix a={{3, 2},
             {4,-5},
             {3, 3}};
   vector b={7,40,3};
//---
   vector x=a.LstSq(b);
//--- проверка, должно быть [5, -4]
   Print("x=", x);
//--- проверка, должно быть [7, 40, 3]
   vector b1=a.MatMul(x);
   Print("b1=",b1);

/*
  x=[5.000000000000002,-4]
  b1=[7.000000000000005,40.00000000000001,3.000000000000005]
*/
```

© 2000-2025, MetaQuotes Ltd.

Inv

Вычисляет мультипликативную обратную матрицу по отношению к квадратной невырожденной матрице методом Жордана-Гаусса.

```
matrix matrix::Inv()
```

Возвращаемое значение

Мультипликативная обратная матрица.

Примечание

Произведением исходной матрицы и обратной матрицы является единичная матрица.

Если хотя бы одна строка (столбец) матрицы нулевая, то обратная матрица не может быть получена.

Если две (или несколько) строки (столбца) матрицы линейно зависимы, то обратная матрица не может быть получена.

Пример:

```
int TestInverse(const int size_m)
  {
   int    i,j,errors=0;
   matrix matrix_a(size_m,size_m);
//--- заполним квадратную матрицу
   MatrixTestFirst(matrix_a);
//--- будем мерить микросекунды
   ulong t1=GetMicrosecondCount();
//--- получим обратную матрицу
   matrix inverse=matrix_a.Inv();
//--- замерим
   ulong t2=GetMicrosecondCount();
//--- проверим правильность
   matrix identity=matrix_a.MatMul(inverse);
//---
   for(i=0; i<size_m; i++)
     {
      for(j=0; j<size_m; j++)
        {
         double value;
         //--- по диагонали должны быть единицы
         if(i==j)
            value=1.0;
         else
            value=0.0;
         if(MathClassify(identity[i][j])>FP_ZERO)
```

© 2000-2025, MetaQuotes Ltd.

```
            errors++;
         else
           {
            if(identity[i][j]!=value)
              {
               double diff=MathAbs(identity[i][j]-value);
               //--- много умножений и делений, поэтому снижаем точность проверки
               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,er
   return(errors);
  }
```

© 2000-2025, MetaQuotes Ltd.

PInv

Вычисляет псевдообратную матрицу методом Мура-Пенроуза.

```
matrix matrix::PInv()
```

Возвращаемое значение

Псевдообратная матрица.

Пример:

```
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;
//--- заполним матрицу
   MatrixTestFirst(matrix_a);
//--- инвертируем
   matrix_inverted=matrix_a.PInv();
//--- проверим правильность
   int errors=0;
//--- A * A+ * A = A   (A+ - псевдообратная матрица от 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":"fai
//---
   return(errors);
  }
```

© 2000-2025, MetaQuotes Ltd.

Машинное обучение

Методы для использования в машинном обучении.

Функция активации в нейронной сети определяет выходное значение нейрона в зависимости от результата взвешенной суммы входов. Выбор функции активации имеет большое влияние на возможности и производительность нейронной сети. В разных частях модели (слоях) могут использоваться различные функции активации.

В MQL5 реализованы не только все известные функции активации, но и их производные. Производные функций позволяют эффективно обновлять параметры модели на основании полученной в процессе обучения ошибки.

Задача обучения нейронной сети заключается в поиске алгоритма, минимизирующего ошибку на обучающей выборке, для чего используется функция потерь (loss function). Значение функции потерь характеризует величину отклонения предсказываемого моделью значения от реального. В зависимости от типа решаемой задачи используются различные функции потерь. Например: для задачи регрессии mean squared error (MSE), для бинарной классификации — binary cross-entropy (BCE).

Функция	Действие
Activation	Вычисляет значения функции активации и записывает в переданный вектор/матрицу
Derivative	Вычисляет значения производной активационной функции и записывает в переданный вектор/матрицу
Loss	Вычисляет значения функции потерь и записывает в переданный вектор/матрицу
LossGradient	Вычисляет вектор или матрицу градиентов функции потерь
RegressionMetric	Вычисляет регрессионную метрику для оценки качества предсказанных данных по отношению к истинным данным
ConfusionMatrix	Вычисляет матрицу ошибок. Метод применяется к вектору предсказанных значений
ConfusionMatrixMultilabel	Вычисляет матрицу ошибок для каждой метки. Метод применяется к вектору предсказанных значений
ClassificationMetric	Вычисляет классификационную метрику для оценки качества предсказанных данных по отношению к истинным данным. Метод применяется к вектору предсказанных значений
ClassificationScore	Вычисляет классификационную метрику для оценки качества предсказанных данных по отношению к истинным данным
PrecisionRecall	Вычисляет значения для построения кривой precision-recall. Данный метод, как и метод ClassificationScore, применяется к вектору истинных значений
ReceiverOperatingC	Вычисляет значения для построения кривой Receiver Operating Characteristic (ROC). Данный метод, как и метод ClassificationScore, применяется к вектору

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
haracteristic	истинных значений

Пример:

Этот пример демонстрирует обучение модели с использованием матричных операций. Модель обучается функции (a + b + c)^2 / (a^2 + b^2 + c^2). На вход подается матрица исходных данных, в которых a, b и c содержатся в отдельных колонках. На выходе модели возвращается результат функции.

```
matrix weights1, weights2, weights3;                // матрицы весовых коэффициентов
matrix output1, output2, result;                    // матрицы выходов нейронных слоев
input int layer1 = 200;                             // размер 1-го скрытого слоя
input int layer2 = 200;                             // размер 2-го скрытого слоя
input int Epochs = 20000;                           // количество эпох обучения
input double lr = 3e-6;                             // коэффициент скорости обучения
input ENUM_ACTIVATION_FUNCTION ac_func = AF_SWISH; // функция активации
//+------------------------------------------------------------------+
//| Функция запуска скрипта                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int train = 1000;    // размер обучающей выборки
   int test = 10;       // размер тестовой выборки
   matrix m_data, m_target;
//--- генерируем обучающую выборку
   if(!CreateData(m_data, m_target, train))
      return;
//--- обучаем модель
   if(!Train(m_data, m_target, Epochs))
      return;
//--- генерируем тестовую выборку
   if(!CreateData(m_data, m_target, test))
      return;
//--- тестирование модели
   Test(m_data, m_target);
  }
//+------------------------------------------------------------------+
//| Метод генерации выборки                                          |
//+------------------------------------------------------------------+
bool CreateData(matrix &data, matrix &target, const int count)
  {
//--- инициализируем матрицы исходных данных и результатов
   if(!data.Init(count, 3) || !target.Init(count, 1))
      return false;
//--- заполняем матрицу исходных данных случайными значениями
```

© 2000-2025, MetaQuotes Ltd.

```
   data.Random(-10, 10);
//--- рассчитываем целевые значения для обучающей выборки
   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;
//--- возвращаем результат
   return true;
  }
//+------------------------------------------------------------------+
//| Метод обучения модели                                            |
//+------------------------------------------------------------------+
bool Train(matrix &data, matrix &target, const int epochs = 10000)
  {
//--- создаем модель
   if(!CreateNet())
      return false;
//--- обучаем модель
   for(int ep = 0; ep < epochs; ep++)
     {
      //--- прямой проход
      if(!FeedForward(data))
         return false;
      PrintFormat("Epoch %d, loss %.5f", ep, result.Loss(target, LOSS_MSE));
      //--- обратный проход и обновление матриц весов
      if(!Backprop(data, target))
         return false;
     }
//--- возвращаем результат
   return true;
  }
//+------------------------------------------------------------------+
//| Метод создания модели                                            |
//+------------------------------------------------------------------+
bool CreateNet()
  {
//--- инициализируем матрицы весов
   if(!weights1.Init(4, layer1) || !weights2.Init(layer1 + 1, layer2) || !weights3.Ini
      return false;
//--- заполняем матрицы весов случайными значениями
   weights1.Random(-0.1, 0.1);
   weights2.Random(-0.1, 0.1);
   weights3.Random(-0.1, 0.1);
//--- возвращаем результат
   return true;
  }
//+------------------------------------------------------------------+
//| Метод прямого прохода                                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool FeedForward(matrix &data)
  {
//--- проверяем размер исходных данных
   if(data.Cols() != weights1.Rows() - 1)
      return false;
//--- вычисляем первый нейронный слой
   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);
//--- вычисяем функцию активации
   if(!output1.Activation(temp, ac_func))
      return false;
//--- вычисляем второй нейронный слой
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   output2 = temp.MatMul(weights2);
//--- вычисляем функцию активации
   if(!output2.Activation(temp, ac_func))
      return false;
//--- вычисляем третий нейронный слой
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   result = temp.MatMul(weights3);
//--- возвращаем результат
   return true;
  }
//+------------------------------------------------------------------+
//| Метод обратного прохода                                          |
//+------------------------------------------------------------------+
bool Backprop(matrix &data, matrix &target)
  {
//--- проверяем размерность матрицы целевых значений
   if(target.Rows() != result.Rows() ||
      target.Cols() != result.Cols())
      return false;
//--- определяем отклонение расчетных значений от целевых
   matrix loss = (target - result) * 2;
//--- проводим градиент до предыдущего слоя
   matrix gradient = loss.MatMul(weights3.Transpose());
//--- обновляем матрицу весов последнего слоя
   matrix temp;
   if(!output2.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
```

© 2000-2025, MetaQuotes Ltd.

```
      return false;
   weights3 = weights3 + temp.Transpose().MatMul(loss) * lr;
//--- корректируем градиент ошибки на производную функции активации
   if(!output2.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- опускаем градиент на слой ниже
   gradient = loss.MatMul(weights2.Transpose());
//--- обновляем матрицу весов 2-го скрытого слоя
   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;
//--- корректируем градиент ошибки на производную функции активации
   if(!output1.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- обновляем матрицу весов 1-го скрытого слоя
   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;
//--- возвращаем результат
   return true;
  }
//+------------------------------------------------------------------+
//| Метод тестирования модели                                        |
//+------------------------------------------------------------------+
bool Test(matrix &data, matrix &target)
  {
//--- прямой проход на тестовых данных
   if(!FeedForward(data))
      return false;
//--- выводим в лог результаты расчета модели и истинные значения
   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]);
//--- возвращаем результат
   return true;
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

Activation

Вычисляет значения функции активации и записывает в переданный вектор/матрицу.

```
bool vector::Activation(
  vector&                    vect_out,      // вектор для получения значений
  ENUM_ACTIVATION_FUNCTION   activation,    // функция активации
   ...                                      // дополнительные параметры
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // матрица для получения значений
  ENUM_ACTIVATION_FUNCTION   activation      // функция активации
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // матрица для получения значений
  ENUM_ACTIVATION_FUNCTION   activation,    // функция активации
  ENUM_MATRIX_AXIS           axis,          // ось
   ...                                      // дополнительные параметры
   );
```

Параметры

vect_out/matrix_out

[out] Вектор или матрица для получения вычисленных значений функции активации.

activation

[in] Функция активации из перечисления ENUM_ACTIVATION_FUNCTION.

axis

[in] Значение из перечисления ENUM_MATRIX_AXIS (AXIS_HORZ — горизонтальная ось, AXIS_VERT — вертикальная ось).

...

[in] Дополнительные параметры, необходимые для некоторых функций активации. Если параметры не указаны, то используются значения по умолчанию.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Дополнительные параметры

Некоторые функции активации принимают дополнительные параметры. Если параметры не указаны, то используются значения по умолчанию

© 2000-2025, MetaQuotes Ltd.

```
AF_ELU  (Exponential Linear Unit)
  double alpha=1.0

Функция активации: if(x>=0) f(x) = x
                   else f(x) = alpha * (exp(x)-1)


AF_LINEAR
  double alpha=1.0
  double beta=0.0

Функция активации: f(x) = alpha*x + beta


AF_LRELU   (Leaky REctified Linear Unit)
  double alpha=0.3

Функция активации: 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

Функция активации: 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

Функция активации: f(x) = x / (1+exp(-x*beta))


AF_TRELU   (Thresholded REctified Linear Unit)
  double theta=1.0

Функция активации: if(x>theta) f(x) = x
                   else f(x) = 0


AF_PRELU   (Parametric REctified Linear Unit)
  double alpha[] - learned array of coeefficients
```

© 2000-2025, MetaQuotes Ltd.

```
Функция активации: if(x[i]>=0) f(x)[i] = x[i]
                   else f(x)[i] = alpha[i] * x[i]
```

Примечание

В искусственных нейронных сетях функция активации нейрона определяет выходной сигнал, который определяется входным сигналом или набором входных сигналов. Выбор функции активации имеет большое влияние на возможности и производительность нейронной сети. В разных частях модели (слоях) могут использоваться различные функции активации.

Примеры использования дополнительных параметров:

```
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("");
```

© 2000-2025, 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);

/*  Результаты
   [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]
*/
```

© 2000-2025, MetaQuotes Ltd.

Derivative

Вычисляет значения производной активационной функции и записывает в переданный вектор/матрицу.

```
bool vector::Derivative(
  vector&                   vect_out,      // вектор для получения значений
  ENUM_ACTIVATION_FUNCTION  activation,    // функция активации
   ...                                     // дополнительные параметры
   );


bool matrix::Derivative(
  matrix&                   matrix_out,    // матрица для получения значений
  ENUM_ACTIVATION_FUNCTION  activation,    // функция активации
   );


bool matrix::Derivative(
  matrix&                   matrix_out,    // матрица для получения значений
  ENUM_ACTIVATION_FUNCTION  activation,    // функция активации
  ENUM_MATRIX_AXIS          axis,          // ось
   ...                                     // дополнительные параметры
   );
```

Параметры

vect_out/matrix_out

[out] Вектор или матрица для получения вычисленных значений производной активационной функции.

activation

[in] Функция активации из перечисления ENUM_ACTIVATION_FUNCTION.

axis

[in] Значение из перечисления ENUM_MATRIX_AXIS (AXIS_HORZ — горизонтальная ось, AXIS_VERT — вертикальная ось).

...

[in] Дополнительные параметры, такие же, как у функций активации. Только некоторые функции активации принимают дополнительные параметры. Если параметры не указаны, то используются значения по умолчанию.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

© 2000-2025, MetaQuotes Ltd.

Производные функций позволяют эффективно обновлять параметры модели на основании полученной в процессе обучения ошибки при обратном распространении ошибки.

© 2000-2025, MetaQuotes Ltd.

Loss

Вычисляет значение функции потерь.

```
double vector::Loss(
  const vector&        vect_true,     // вектор истинных значений
  ENUM_LOSS_FUNCTION   loss,          // функция потерь
   ...                                // дополнительный параметр
   );


double matrix::Loss(
  const matrix&        matrix_true,   // матрица истинных значений
  ENUM_LOSS_FUNCTION   loss,          // функция потерь
   );


double matrix::Loss(
  const matrix&        matrix_true,   // матрица истинных значений
  ENUM_LOSS_FUNCTION   loss,          // функция потерь
  ENUM_MATRIX_AXIS     axis,          // ось
   ...                                // дополнительный параметр
   );
```

Параметры

vect_true/matrix_true

[in] Вектор или матрица истинных значений.

loss

[in] Функция потерь из перечисления ENUM_LOSS_FUNCTION.

axis

[in] Значение из перечисления ENUM_MATRIX_AXIS (AXIS_HORZ — горизонтальная ось, AXIS_VERT — вертикальная ось).

...

[in] Дополнительный параметр delta может быть только у функции потерь Юбера (LOSS_HUBER)

Возвращаемое значение

Значение double.

Как используется параметр delta в функции потерь Юбера (LOSS_HUBER)

```
   double delta = 1.0;
   double error = fabs(y - x);
   if(error<delta)
      loss = 0.5 * error^2;
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      loss = 0.5 * delta^2 + delta * (error - delta);
```

Примечание

Задача обучения нейронной сети заключается в поиске коэффициентов, минимизирующих ошибку на обучающей выборке, для чего используется функция потерь (loss function).

Значение функции потерь характеризует величину отклонения предсказываемого моделью значения от реального.

В зависимости от типа решаемой задачи используются различные функции потерь. Например: для задачи регрессии mean squared error (MSE), для бинарной классификации — binary cross-entropy (BCE).

Пример вызова функции потерь Юбера:

```
   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);

/* Результат
   0.155
   0.15125
*/
```

© 2000-2025, MetaQuotes Ltd.

LossGradient

Вычисляет вектор или матрицу градиентов функции потерь.

```
vector vector::LossGradient(
  const vector&       vect_true,     // вектор истинных значений
  ENUM_LOSS_FUNCTION  loss,          // тип функции потерь
   ...                               // дополнительный параметр
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,   // матрица истинных значений
  ENUM_LOSS_FUNCTION  loss,          // функция потерь
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,   // матрица истинных значений
  ENUM_LOSS_FUNCTION  loss,          // функция потерь
  ENUM_MATRIX_AXIS    axis,          // ось
   ...                               // дополнительный параметр
   );
```

Параметры

vect_true/matrix_true

[in] Вектор или матрица истинных значений.

loss

[in] Функция потерь из перечисления ENUM_LOSS_FUNCTION.

axis

[in] Значение из перечисления ENUM_MATRIX_AXIS (AXIS_HORZ — горизонтальная ось, AXIS_VERT — вертикальная ось).

...

[in] Дополнительный параметр delta может быть только у функции потерь Юбера (LOSS_HUBER)

Возвращаемое значение

Вектор или матрица значений градиентов функции потерь. Градиент — это частная производная по dx (x — предсказанное значение) от функции потерь в данной точке.

Примечание

Градиенты используются в нейронных сетях для корректировки значений матрицы весов при обратном распространении ошибки при обучении модели.

© 2000-2025, MetaQuotes Ltd.

Задача обучения нейронной сети заключается в поиске коэффициентов, минимизирующих ошибку на обучающей выборке, для чего используется функция потерь (loss function).

В зависимости от типа решаемой задачи используются различные функции потерь. Например: для задачи регрессии mean squared error (MSE), для бинарной классификации — binary cross-entropy (BCE).

Пример вычисления градиентов функции потерь

```
   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);

/* Результат
   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]]
*/
```

© 2000-2025, MetaQuotes Ltd.

RegressionMetric

Вычисляет регрессионную метрику для оценки качества предсказанных данных по отношению к истинным данным

```
double vector::RegressionMetric(
   const vector&           vector_true,   // вектор истинных значений
   ENUM_REGRESSION_METRIC  metric         // тип метрики
   );

double matrix::RegressionMetric(
   const matrix&           matrix_true,   // матрица истинных значений
   ENUM_REGRESSION_METRIC  metric         // тип метрики
);

vector matrix::RegressionMetric(
   const matrix&           matrix_true,   // матрица истинных значений
   ENUM_REGRESSION_METRIC  metric,        // тип метрики
   int                     axis           // ось
   );
```

Параметры

vector_true/matrix_true

[in] Вектор или матрица истинных значений.

metric

[in] Тип метрики из перечисления ENUM_REGRESSION_METRIC.

axis

[in] Ось. 0 — горизонтальная ось, 1 — вертикальная ось.

Возвращаемое значение

Рассчитанная метрика — оценка качества предсказанных данных по отношению к истинным данным.

Примечание

- REGRESSION_MAE — среднее абсолютных разностей между предсказанными значениями и соответствующими истинными значениями
- REGRESSION_MSE — среднее квадратов разностей между предсказанными значениями и соответствующими истинными значениями
- REGRESSION_RMSE — квадратный корень из MSE
- REGRESSION_R2 - 1 — MSE(регрессия) / MSE(среднее)
- REGRESSION_MAPE — средняя абсолютная ошибка (MAE) в процентах
- REGRESSION_MSPE — среднеквадратичная ошибка (MSE) в процентах
- REGRESSION_RMSLE — RMSE, рассчитанная в логарифмическом масштабе

© 2000-2025, MetaQuotes Ltd.

Пример:

```
   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);

   /* Результат
   mae=0.375
   mse=0.5
   r2=0.9486081370449679
   */
```

© 2000-2025, MetaQuotes Ltd.

ConfusionMatrix

Вычисляет матрицу ошибок. Метод применяется к вектору предсказанных значений.

```
matrix vector::ConfusionMatrix(
   const vector&        vect_true      // вектор истинных значений
   );


matrix vector::ConfusionMatrix(
   const vector&        vect_true,     // вектор истинных значений
   uint                 label          // значение метки
   );
```

Параметры

vect_true

[in] Вектор истинных значений.

эпохи

[in] Значение метки для вычисления матрицы ошибок.

Возвращаемое значение

Матрица ошибок. Если значение метки не указано, то возвращается многоклассовая матрица ошибок, где каждая метка сопоставляется с каждой другой меткой по отдельности. Если указано значение метки, то возвращается матрица размером 2 x 2, где указанная метка считается положительной, все остальные метки - отрицательные (ovr, one vs rest).

Примечание

Матрица ошибок C такова, что Cij равен количеству наблюдений, которые находятся в группе i, а также, по прогнозам, - в группе j. Таким образом, в бинарной классификации количество истинно отрицательных результатов (TN) равно C00, ложноотрицательных результатов (FN) — C10, истинно положительных результатов (TP) — C11 и ложноположительных результатов (FP) — C01.

То есть эту матрицу можно графически представить таким образом:

TN	FP
FN	TP

Размеры вектора истинных значений и вектора предсказанных значений должны быть одинаковыми.

Пример:

```
   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};
```

© 2000-2025, MetaQuotes Ltd.

```
   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 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]]
*/
```

© 2000-2025, MetaQuotes Ltd.

ConfusionMatrixMultilabel

Вычисляет матрицу ошибок для каждой метки. Метод применяется к вектору предсказанных значений.

```
uint vector::ConfusionMatrixMultiLabel(
   const vector&       vect_true,     // вектор истинных значений
   matrix&             confusions[]   // массив вычисленных матриц ошибок
   );
```

Параметры

vect_true

[in] Вектор истинных значений.

confusions

[out] Массив матриц 2 x 2 с вычисленными матрицами ошибок по каждой метке.

Возвращаемое значение

Размер массива вычисленных матриц ошибок. В случае неудачи возвращается 0

Примечание

Результатный массив может быть динамический или статический. Если массив статический, то он должен быть размера не меньше, чем количество классов.

Размеры вектора истинных значений и вектора предсказанных значений должны быть одинаковыми.

Пример:

```
   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]]
```

© 2000-2025, MetaQuotes Ltd.

```
   [[25,2]
    [2,1]]
   [[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]]
*/
```

© 2000-2025, MetaQuotes Ltd.

ClassificationMetric

Вычисляет классификационную метрику для оценки качества предсказанных данных по отношению к истинным данным. Метод применяется к вектору предсказанных значений.

```
vector vector::ClassificationMetric(
   const vector&               vect_true,      // вектор истинных значений
   ENUM_CLASSIFICATION_METRIC metric          // тип метрики
   );


vector vector::ClassificationMetric(
   const vector&               vect_true,      // вектор истинных значений
   ENUM_CLASSIFICATION_METRIC metric          // тип метрики
   ENUM_AVERAGE_MODE           mode           // режим усреднения
   );
```

Параметры

vect_true

[in] Вектор истинных значений.

metric

[in] Тип метрики из перечисления ENUM_CLASSIFICATION_METRIC. Используются значения кроме CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION и CLASSIFICATION_ROC_AUC (которые используются в методе ClassificationScore).

mode

[in] Режим усреднения из перечисления ENUM_AVERAGE_MODE. Используется для метрик CLASSIFICATION_F1, CLASSIFICATION_JACCARD, CLASSIFICATION_PRECISION и CLASSIFICATION_RECALL.

Возвращаемое значение

Вектор, содержащий рассчитанную метрику. В случае режима усреднения AVERAGE_NONE вектор содержит значения метрик для каждого класса без усреднения. (Например, для бинарной классификации это будет 2 метрики для false и true соответственно).

Примечание про режимы усреднения

AVERAGE_BINARY имеет смысл только для бинарной классификации.

AVERAGE_MICRO — рассчитать метрики глобально путем подсчета общего количества истинных, ложноотрицательных и ложноположительных результатов.

AVERAGE_MACRO — рассчитать метрики для каждой метки и найти их невзвешенное среднее значение. При этом не учитывается дисбаланс меток.

AVERAGE_WEIGHTED — рассчитать метрики для каждой метки и найти их средневзвешенное значение по поддержке (количеству истинных экземпляров для каждой метки). Такой расчет меняет macro с учетом дисбаланса меток, что может привести к F-мере, которая не находится между точностью и полнотой.

© 2000-2025, MetaQuotes Ltd.

Пример:

```
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_WEI
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,AVER
Print("jaccard_micro=",jaccard_micro);
vector jaccard_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVER
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);
```

© 2000-2025, 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=[0.8571428571428571,1,0.3333333333333333,0.6666666666666666,0.6666666666666

  jaccard_micro=[0.5]
  jaccard_macro=[0.4921428571428572]
  jaccard_weighted=[0.5056349206349205]
  jaccard_none=[0.75,1,0.2,0.5,0.5,0.6666666666666666,0.3333333333333333,0.4,0,0.5714

  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]
*/
```

© 2000-2025, MetaQuotes Ltd.

ClassificationScore

Вычисляет классификационную метрику для оценки качества предсказанных данных по отношению к истинным данным.

В отличие от других методов, относящихся к разделу "Машинное обучение", данный метод применяется к вектору истинных значений, а не к вектору предсказанных значений.

```
vector vector::ClassificationScore(
   const matrix&               pred_scores,    // матрица, содержащая распределение вер
   ENUM_CLASSIFICATION_METRIC metric          // тип метрики
   ENUM_AVERAGE_MODE           mode           // режим усреднения
   );


vector vector::ClassificationScore(
   const matrix&               pred_scores,    // матрица, содержащая распределение вер
   ENUM_CLASSIFICATION_METRIC metric          // тип метрики
   int                         param          // дополнительный параметр
   );
```

Параметры

pred_scores

[in] Матрица, содержащая набор горизонтальных векторов с вероятностями по каждому классу. Количество строк матрицы должно соответствовать размеру вектора истинных значений.

metric

[in] Тип метрики из перечисления ENUM_CLASSIFICATION_METRIC. Используются значения CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION и CLASSIFICATION_ROC_AUC.

mode

[in] Режим усреднения из перечисления ENUM_AVERAGE_MODE. Используется для метрик CLASSIFICATION_AVERAGE_PRECISION и CLASSIFICATION_ROC_AUC.

param

[in] Для метрики CLASSIFICATION_TOP_K_ACCURACY вместо режима усреднения нужно указать целочисленное значение K.

Возвращаемое значение

Вектор, содержащий рассчитанную метрику. В случае режима усреднения AVERAGE_NONE вектор содержит значения метрик для каждого класса без усреднения. (Например, для бинарной классификации это будет 2 метрики для false и true соответственно).

Примечание про режимы усреднения

© 2000-2025, MetaQuotes Ltd.

AVERAGE_BINARY имеет смысл только для бинарной классификации.

AVERAGE_MICRO — рассчитать метрики глобально, рассматривая каждый элемент матрицы индикаторов меток как метку. Под матрицей индикаторов меток подразумевается матрица с набором вероятностей по каждой метке.

AVERAGE_MACRO — рассчитать метрики для каждой метки и найти их невзвешенное среднее значение. При этом не учитывается дисбаланс меток.

AVERAGE_WEIGHTED — рассчитать метрики для каждой метки и найти их средневзвешенное значение по поддержке (количеству истинных экземпляров для каждой метки).

Примечание

Для двоичной классификации на вход можно подавать не только матрицу n x 2, где в первой колонке — вероятности для отрицательной метки, а во второй — вероятности для положительной, но и матрицу, состоящую из одной колонки с положительными вероятностями. Это связано с тем, что модели двоичной классификации могут возвращать как две вероятности, так и одну вероятность для положительной метки.

Пример:

```
   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
```

© 2000-2025, MetaQuotes Ltd.

```
                     {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
                     {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_PREC
   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},
```

© 2000-2025, MetaQuotes Ltd.

```
                        {0.4, 0.6},
                        {0.1, 0.9},
                        {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]
*/
```

© 2000-2025, MetaQuotes Ltd.

PrecisionRecall

Вычисляет значения для построения кривой precision-recall. Данный метод, как и метод ClassificationScore, применяется к вектору истинных значений.

```
bool vector::PrecisionRecall(
   const matrix&                 pred_scores,   // матрица, содержащая распределение
   const ENUM_ENUM_AVERAGE_MODE  mode           // режим усреднения
   matrix&                       precision,     // рассчитанные значения precision для
   matrix&                       recall,        // рассчитанные значения recall для ка
   matrix&                       thresholds,    // пороговые значения, отсортированные
   );
```

Параметры

pred_scores

[in] Матрица, содержащая набор горизонтальных векторов с вероятностями по каждому классу. Количество строк матрицы должно соответствовать размеру вектора истинных значений.

mode

[in] Режим усреднения из перечисления ENUM_AVERAGE_MODE. Используются только AVERAGE_NONE, AVERAGE_BINARY и AVERAGE_MICRO.

precision

[out] Матрица с рассчитанными значениями кривой precision. При отсутствии усреднения (AVERAGE_NONE) количество строк в матрице соответствует количеству классов модели. Количество колонок соответствует размеру вектора истинных значений (или количеству строк в матрице распределений вероятностей pred_score). В случае микроусреднения количество строк в матрице соответствует общему количеству пороговых значений за исключением дублей.

recall

[out] Матрица с рассчитанными значениями кривой recall.

threshold

[out] Матрица пороговых значений, полученная путём сортировки матрицы вероятностей

Примечание

Смотрите примечания для метода ClassificationScore.

Пример

Пример сбора статистики модели mnist.onnx (точность 99%).

```
//--- data for classification metrics
   vectorf y_true(images);
   vectorf y_pred(images);
   matrixf y_scores(images,10);
//--- input-output
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixf image(28,28);
   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);
     }    }
```

Вычисление точности

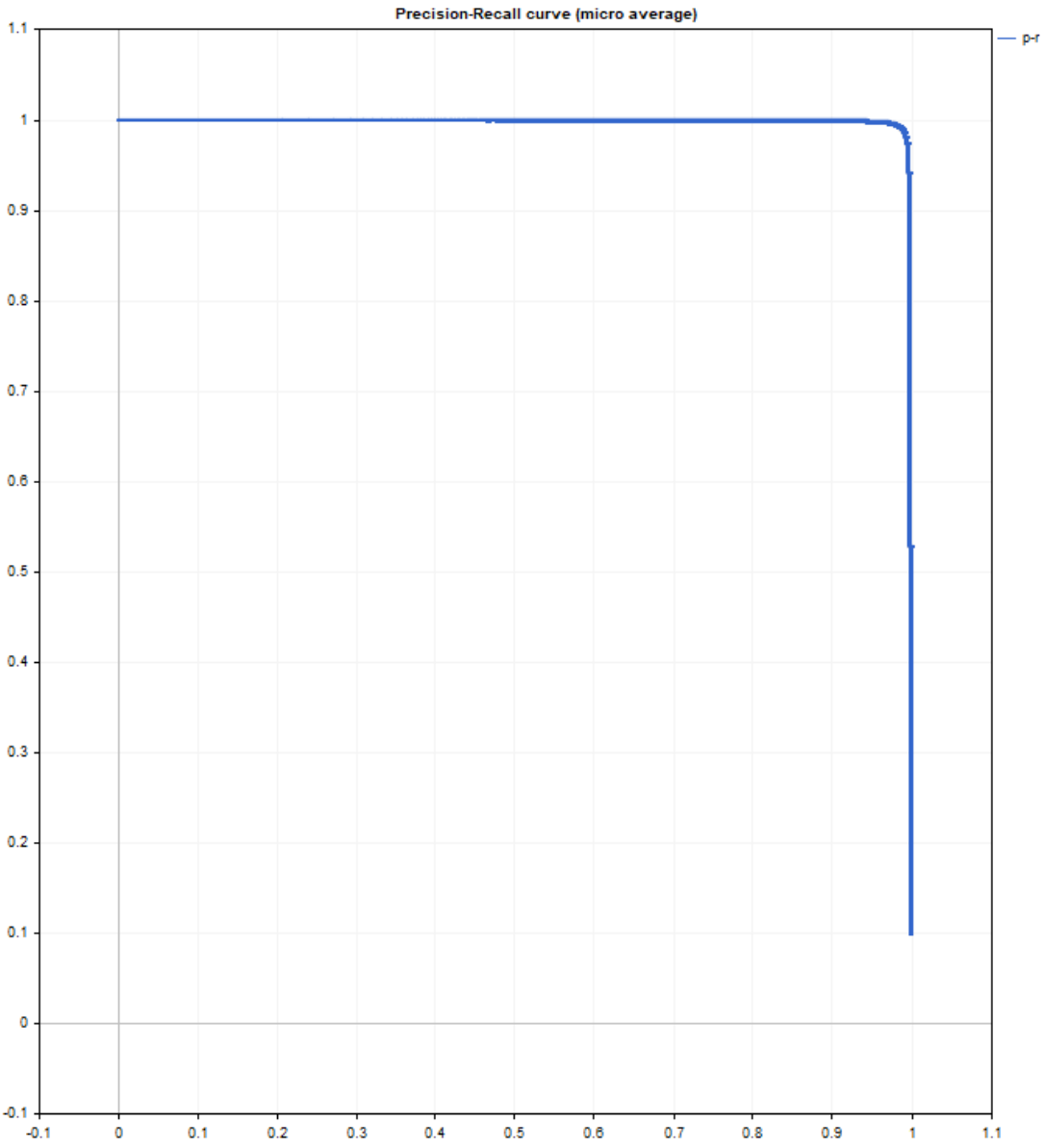
```
   vectorf accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
   PrintFormat("accuracy=%f",accuracy[0]);

accuracy=0.989000
```

Пример вывода графиков precision-recall, где на оси y отложены значения precision, на оси x - значения recall. А также вывод отдельных графиков precision и recall, где на оси x отложены пороговые значения

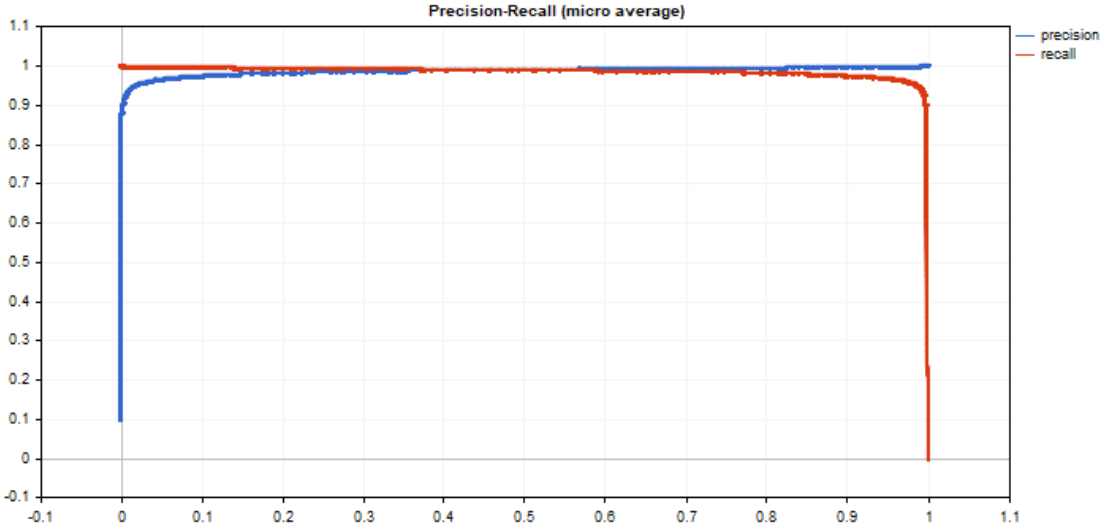
```
   if(y_true.PrecisionRecall(y_scores,AVERAGE_MICRO,mat_precision,mat_recall,mat_thres
     {
      double precision[],recall[],thres[];
      ArrayResize(precision,mat_thres.Cols());
      ArrayResize(recall,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      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
     }
```

© 2000-2025, MetaQuotes Ltd.

Результат:



© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

ReceiverOperatingCharacteristic

Вычисляет значения для построения кривой Receiver Operating Characteristic (ROC). Данный метод, как и метод ClassificationScore, применяется к вектору истинных значений.

```
bool vector::ReceiverOperatingCharacteristic(
   const matrix&                 pred_scores,   // матрица, содержащая распределение в
   const ENUM_ENUM_AVERAGE_MODE  mode           // режим усреднения
   matrix&                       fpr,           // рассчитанные значения false positiv
   matrix&                       tpr,           // рассчитанные значения true positive
   matrix&                       thresholds,    // пороговые значения, отсортированные
   );
```

Параметры

pred_scores

[in] Матрица, содержащая набор горизонтальных векторов с вероятностями по каждому классу. Количество строк матрицы должно соответствовать размеру вектора истинных значений.

mode

[in] Режим усреднения из перечисления ENUM_AVERAGE_MODE. Используются только AVERAGE_NONE, AVERAGE_BINARY и AVERAGE_MICRO.

fpr

[out] Матрица с рассчитанными значениями кривой false positive rate. В случае с отсутствием усреднения (AVERAGE_NONE) количество строк в матрице соответствует количеству классов модели. Количество колонок соответствует размеру вектора истинных значений (или количеству строк в матрице распределений вероятностей pred_score). В случае микроусреднения количество строк в матрице соответствует общему количеству пороговых значений за исключением дублей.

tpr

[out] Матрица с рассчитанными значениями кривой true positive rate.

threshold

[out] Матрица пороговых значений, полученная путём сортировки матрицы вероятностей

Примечание

Смотрите примечания для метода ClassificationScore.

Пример

Пример вывода графиков ROC, где на оси y отложены значения tpr, на оси x - значения fpr. А также вывод отдельных графиков fpr и tpr, где на оси x отложены пороговые значения.

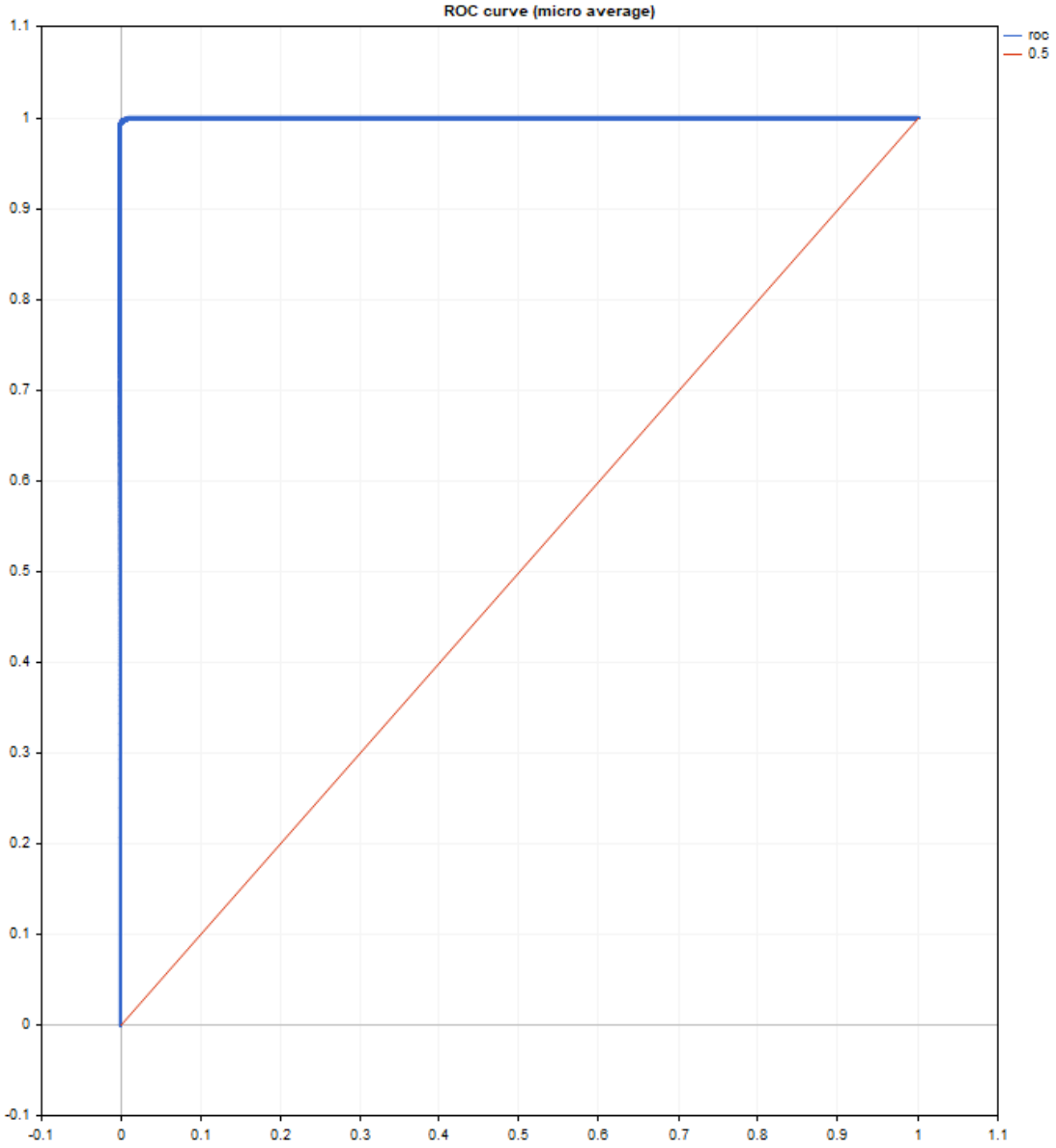
```
   matrixf mat_thres;
   matrixf mat_fpr;
   matrixf mat_tpr;
```

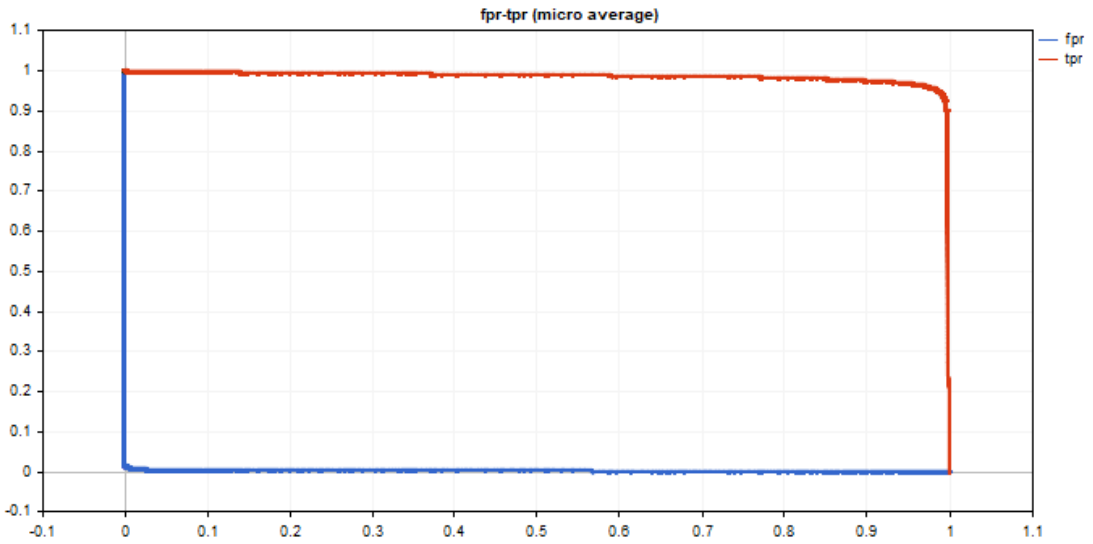
© 2000-2025, MetaQuotes Ltd.

```
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);
  }
```

Результат:

© 2000-2025, MetaQuotes Ltd.





© 2000-2025, MetaQuotes Ltd.

Код вывода графиков элементарный и основан на стандартной библиотеке <Graphics/Graphic.mqh>.

Использованы данные тестирования модели mnist.onnx, код представлен в описании метода PrecisionRecall.

ROC AUC близок к идеальному.

```
roc auc score micro = [0.99991]
```

© 2000-2025, MetaQuotes Ltd.

Методы OpenBLAS

OpenBLAS — это высокопроизводительная библиотека линейной алгебры с открытым исходным кодом, реализующая BLAS (Basic Linear Algebra Subprograms) и некоторые функции LAPACK. OpenBLAS нацелен на повышение производительности вычислений, особенно в операциях с матрицами и векторными вычислениями, которые важны в научных и инженерных задачах, таких как машинное обучение, численные методы и симуляции.

Основные особенности OpenBLAS:

- Поддержка многопоточности: OpenBLAS может эффективно использовать несколько процессорных ядер для параллельных вычислений, что значительно ускоряет выполнение операций на многопроцессорных системах.
- Оптимизация под архитектуры процессоров: OpenBLAS включает оптимизированные сборки для различных процессоров, таких как Intel, AMD, ARM и других. Библиотека автоматически определяет характеристики процессора и выбирает наиболее подходящие реализации функций.
- Поддержка широкого спектра операций BLAS: OpenBLAS реализует основные функции BLAS, такие как операции с векторами (например, векторное сложение и скалярное произведение), матрицами (умножение) и векторно-матричные операции.
- Совместимость с LAPACK: Библиотека поддерживает ряд функций LAPACK (Linear Algebra PACKage), которые нужны для более сложных операций линейной алгебры, таких как решения систем линейных уравнений, вычисление собственных значений матриц и другие.
- Высокая производительность: По сравнению с другими библиотеками BLAS, OpenBLAS часто показывает лучшие результаты благодаря ручной оптимизации под конкретные процессорные архитектуры.

Применение

OpenBLAS широко используется в приложениях, связанных с численными вычислениями, таких как:

- Обучение нейронных сетей и другие задачи машинного обучения.
- Научные вычисления, например, моделирование физических процессов.
- Обработка больших объемов данных и их анализ.

Библиотека интегрирована во многие популярные научные программные пакеты, такие как NumPy, SciPy и TensorFlow, которые зависят от высокопроизводительных операций линейной алгебры для эффективной работы.

OpenBLAS — отличный выбор для тех, кто ищет решение с открытым исходным кодом для высокопроизводительных вычислений, особенно при работе с большими матрицами и векторами.

Функция	Действие
SingularValueDecompositionDC	Singular Value Decomposition, "divide and conquer" алгоритм. Данный алгоритм считается наиболее быстрым среди других алгоритмов SVD (lapack-функция GESDD).

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
SingularValueDecompositionQR	Singular Value Decomposition, QR алгоритм. Данный алгоритм считается классическим алгоритмом SVD (lapack-функция GESVD).
SingularValueDecompositionQRPivot	Singular Value Decomposition, алгоритм QR with pivoting (lapack-функция GESVDQ).
SingularValueDecompositionBisect	Singular Value Decomposition, алгоритм bisection (lapack-функция GESVDX).
SingularValueDecompositionJacobiHigh	Singular Value Decomposition, алгоритм Jacobi high level (lapack-функция GEJSV).
SingularValueDecompositionJacobiLow	Singular Value Decomposition, алгоритм Jacobi low level (lapack-функция GESVJ). В некоторых случаях вычисляет малые сингулярные значения и их сингулярные векторы гораздо более точно, чем другие подпрограммы SVD.
SingularValueDecompositionBidiagDC	Singular Value Decomposition, алгоритм divide and conquer для бидиагональной матрицы (lapack-функция BDSDC).
SingularValueDecompositionBidiagBisect	Singular Value Decomposition, алгоритм bisection для бидиагональных матриц (lapack-функция BDSVDX).
EigenSolver	Вычисление собственных значений и собственных векторов обычной квадратной матрицы классическим алгоритмом (lapack-функция GEEV).
EigenSolver2	Вычисление обобщённых собственных значений и собственных векторов для пары обычных квадратных матриц (lapack-функция GGEV).
EigenSolverX	Вычисление собственных значений и собственных векторов обычной квадратной матрицы в режиме эксперта (Expert mode), то есть с возможностью влиять на алгоритм расчёта и возможностью получить сопутствующие расчётные данные (lapack-функция GEEVX).
EigenSolverShur	Вычисление собственных значений, верхнетреугольной матрицы в форме Шура и матрицы векторов Шура (lapack-

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
	функция GEES). Смотри Разложение Шура.
EigenSymmetricDC	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма "divide and conquere" (lapack-функции SYEVD, HEEVD).
EigenSymmetricQR	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи классического алгоритма QR (lapack-функции SYEV, HEEV).
EigenSymmetricRobust	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма Multiple Relatively Robust Representations, MRRR (lapack-функции SYEVR, HEEVR).
EigenSymmetricBisect	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма bisection (lapack-функции SYEVX, HEEVX).
SingularSpectrumAnalysisSpectrum	Функция-метод расчёта относительных вкладов спектральных компонент по собственным значениям.
SingularSpectrumAnalysisForecast	Функция-метод расчёта реконструированных и прогнозируемых данных с использованием спектральных компонент входного временного ряда.
SingularSpectrumAnalysisReconstructComponents	Функция-метод расчёта реконструированных компоненент входного временного ряда и их вклады.
SingularSpectrumAnalysisReconstructSeries	Функция-метод расчёта реконструированного временного ряда с использованиеми первых component_count компонент.

© 2000-2025, MetaQuotes Ltd.

Разложение по сингулярным значениям

В разделе "Анализ сингулярного спектра" (Singular Spectrum Analysis, SSA) содержит функции для работы с временными рядами, позволяющие выделить основные компоненты данных, восстановить временные ряды, выполнить прогнозирование будущих значений и анализировать спектральные компоненты

Функция	Действие
SingularValueDecompositionDC	Singular Value Decomposition — разложение по сингулярным числам, алгоритм "разделяй и властвуй" (divide and conquer). Данный алгоритм считается наиболее быстрым среди других алгоритмов SVD (lapack-функция GESDD).
SingularValueDecompositionQR	Singular Value Decomposition, QR алгоритм. Данный алгоритм считается классическим алгоритмом SVD (lapack-функция GESVD).
SingularValueDecompositionQRPivot	Singular Value Decomposition, алгоритм QR with pivoting (lapack-функция GESVDQ).
SingularValueDecompositionBisect	Singular Value Decomposition — разложение по сингулярным значениям, алгоритм bisection (lapack-функция GESVDX).
SingularValueDecompositionJacobiHigh	Singular Value Decomposition — разложение по сингулярным значениям, алгоритм Jacobi high level (lapack-функция GEJSV).
SingularValueDecompositionJacobiLow	Singular Value Decomposition, алгоритм Jacobi low level (lapack-функция GESVJ). В некоторых случаях вычисляет малые сингулярные значения и их сингулярные векторы гораздо более точно, чем другие подпрограммы SVD.
SingularValueDecompositionBidiagDC	Singular Value Decomposition, алгоритм divide and conquer для бидиагональной матрицы (lapack-функция BDSDC).
SingularValueDecompositionBidiagBisect	Singular Value Decomposition, алгоритм bisection для бидиагональных матриц (lapack-функция BDSVDX).
SingularValueDecompositionBidiagQR	Вычисление сингулярного разложения общей матрицы, предварительно приведенной к бидиагональному виду методом ReduceToBidiagonal. Lapack-функция BDSQR.

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionDC

Singular Value Decomposition — разложение по сингулярным числам, алгоритм "разделяй и властвуй" (divide and conquer). Данный алгоритм считается наиболее быстрым среди других алгоритмов SVD (lapack-функция GESDD).

Вычисления для типа matrix<double>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z      jobz,     // способ вычисления левых и правых векторов
   vector&         S,        // вектор рассчитанных сингулярных значений
   matrix&         U,        // матрица рассчитанных левых векторов U
   matrix&         VT        // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z      jobz,     // способ вычисления левых и правых векторов
   vectorf&        S,        // вектор рассчитанных сингулярных значений
   matrixf&        U,        // матрица рассчитанных левых векторов U
   matrixf&        VT        // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z      jobz,     // способ вычисления левых и правых векторов
   vector&         S,        // вектор рассчитанных сингулярных значений
   matrixc&        U,        // матрица рассчитанных левых векторов U
   matrixc&        VT        // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z      jobz,                  // способ вычисления левых и правых векторо
   vectorf&        singular_values,       // вектор рассчитанных сингулярных значений
   matrixcf&       U,                     // матрица рассчитанных левых векторов U
   matrixcf&       VT                     // транспонированная матрица правых векторо
   );
```

Параметры

jobz

[in] Значение из перечисления ENUM_SVD_Z, определяющее способ вычисления левых и правых сингулярных векторов.

S

[out] Вектор сингулярных значений.

U

© 2000-2025, MetaQuotes Ltd.

[out] Матрица левых сингулярных векторов.

VT

[out] Матрица правых сингулярных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значения параметра jobz.

При значении SVDZ_N левые и правые вектора не рассчитываются. Рассчитываются только сингулярные значения.

При значении SVDZ_A рассчитываются полные матрицы векторов U и VT.

При значении SVDZ_S рассчитываются усечённые матрицы векторов U и VT.

ENUM_SVD_Z

Перечисление, определяющее способ вычисления левых и правых сингулярных векторов.

Идентификатор	Описание
SVDZ_N	Столбцы U или строки VT не вычисляются
SVDZ_A	Все M столбцов U или все N столбцов VT возвращаются в массивах U и VT
SVDZ_S	Первые min(M,N) столбцов U или первые min(M,N) столбцов VT возвращаются в массивах U и VT

Смотри также

SingularValueDecompositionQR, SingularValueDecompositionQRPivot

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionQR

Singular Value Decomposition, QR алгоритм. Данный алгоритм считается классическим алгоритмом SVD (lapack-функция GESVD).

Вычисления для типа matrix<double>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,     // способ вычисления левых векторов
   ENUM_SVD_Z      jobv,     // способ вычисления правых векторов
   vector&         S,        // вектор рассчитанных сингулярных значений
   matrix&         U,        // матрица рассчитанных левых векторов U
   matrix&         VT        // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,     // способ вычисления левых векторов
   ENUM_SVD_Z      jobv,     // способ вычисления правых векторов
   vectorf&        S,        // вектор рассчитанных сингулярных значений
   matrixf&        U,        // матрица рассчитанных левых векторов U
   matrixf&        VT        // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,     // способ вычисления левых векторов
   ENUM_SVD_Z      jobv,     // способ вычисления правых векторов
   vector&         S,        // вектор рассчитанных сингулярных значений
   matrixc&        U,        // матрица рассчитанных левых векторов U
   matrixc&        VT        // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,                  // способ вычисления левых векторов
   ENUM_SVD_Z      jobv,                  // способ вычисления правых векторов
   vectorf&        singular_values,       // вектор рассчитанных сингулярных значений
   matrixcf&       u,                     // матрица рассчитанных левых векторов U
   matrixcf&       vt                     // транспонированная матрица правых векторо
   );
```

Параметры

jobu

[in] Значение из перечисления ENUM_SVD_Z, определяющее способ вычисления левых сингулярных векторов.

jobv

© 2000-2025, MetaQuotes Ltd.

[in] Значение из перечисления ENUM_SVD_Z, определяющее способ вычисления правых сингулярных векторов.

S

[out] Вектор сингулярных значений.

U

[out] Матрица левых сингулярных векторов.

VT

[out] Матрица правых сингулярных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значений параметров jobu и jobv.

При значении SVDZ_N левые (jobu) или правые (jobv) вектора не рассчитываются. Сингулярные значения рассчитываются всегда.

При значении SVDZ_A рассчитываются полные матрицы векторов U (jobu) или VT (jobv).

При значении SVDZ_S рассчитываются усечённые матрицы векторов U (jobu) или VT (jobv).

ENUM_SVD_Z

Перечисление, определяющее способ вычисления левых и правых сингулярных векторов.

Идентификатор	Описание
SVDZ_N	Столбцы U или строки VT не вычисляются
SVDZ_A	Все M столбцов U или все N столбцов VT возвращаются в массивах U и VT
SVDZ_S	Первые min(M,N) столбцов U или первые min(M,N) столбцов VT возвращаются в массивах U и VT

Смотри также

SingularValueDecompositionDC, SingularValueDecompositionQRPivot

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionQRPivot

Singular Value Decomposition, алгоритм QR with pivoting (lapack-функция GESVDQ).

Вычисления для типа matrix<double>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // уровень точности расчёта
   ENUM_SVDQRP_P   jobp,     // использование переворота строк для расчёта
   ENUM_SVDQRP_R   jobr,     // использование треугольной матрицы R для расчёта
   ENUM_SVDQRP_U   jobu,     // способ вычисления левых векторов
   ENUM_SVDQRP_V   jobv,     // способ вычисления правых векторов
   vector&         S,        // вектор рассчитанных сингулярных значений
   matrix&         U,        // матрица рассчитанных левых векторов U
   matrix&         VT        // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // уровень точности расчёта
   ENUM_SVDQRP_P   jobp,     // использование переворота строк для расчёта
   ENUM_SVDQRP_R   jobr,     // использование треугольной матрицы R для расчёта
   ENUM_SVDQRP_U   jobu,     // способ вычисления левых векторов
   ENUM_SVDQRP_V   jobv,     // способ вычисления правых векторов
   vectorf&        S,        // вектор рассчитанных сингулярных значений
   matrixf&        U,        // матрица рассчитанных левых векторов U
   matrixf&        VT        // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // уровень точности расчёта
   ENUM_SVDQRP_P   jobp,     // использование переворота строк для расчёта
   ENUM_SVDQRP_R   jobr,     // использование треугольной матрицы R для расчёта
   ENUM_SVDQRP_U   jobu,     // способ вычисления левых векторов
   ENUM_SVDQRP_V   jobv,     // способ вычисления правых векторов
   vector&         S,        // вектор рассчитанных сингулярных значений
   matrixc&        U,        // матрица рассчитанных левых векторов U
   matrixc&        VT        // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,                  // уровень точности расчёта
   ENUM_SVDQRP_P   jobp,                  // использование переворота строк для расчё
   ENUM_SVDQRP_R   jobr,                  // использование треугольной матрицы R для
   ENUM_SVDQRP_U   jobu,                  // способ вычисления левых векторов
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_SVDQRP_V   jobv,                  // способ вычисления правых векторов
   vectorf&        singular_values,       // вектор рассчитанных сингулярных значений
   matrixcf&       u,                     // матрица рассчитанных левых векторов U
   matrixcf&       vt                     // транспонированная матрица правых векторо
   );
```

Параметры

joba

[in] Значение из перечисления ENUM_SVDQRP_A, определяющее уровень точности расчёта SVD.

jobp

[in] Значение из перечисления ENUM_SVDQRP_P, определяющее использование переворота строк в процессе расчёта.

jobr

[in] Значение из перечисления ENUM_SVDQRP_R, определяющее необходимость транспонирования треугольной матрицы R, полученной в результате начальной QR-факторизации.

jobu

[in] Значение из перечисления ENUM_SVDQRP_U, определяющее способ вычисления левых сингулярных векторов.

jobv

[in] Значение из перечисления ENUM_SVDQRP_V, определяющее способ вычисления правых сингулярных векторов.

S

[out] Вектор сингулярных значений.

U

[out] Матрица левых сингулярных векторов.

VT

[out] Матрица правых сингулярных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Количество строк матрицы не должно быть меньше количества столбцов.

Если рассчитываются и левые, и правые сингулярные векторы, то при установленном значении jobu=SVDQRPU_R значение jobv должно быть установлено в SVDQRPV_R.

ENUM_SVDQRP_A

Перечисление, определяющее уровень точности расчёта SVD.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
SVDQRP_A	Запрошенная точность соответствует обратной ошибке, ограниченной эпсилоном. Это агрессивный уровень усечения.
SVDQRPA_M	Аналогично SVDQRP_A, но усечение более мягкое. Это средний уровень усечения.
SVDQRPA_H	Требуется высокая точность.

ENUM_SVDQRP_P

Перечисление, определяющее использование переворота строк в процессе расчёта.

Идентификатор	Описание
SVDQRPP_P	Строки A упорядочены в порядке убывания. Рекомендуется для числовой надежности.
SVDQRPP_N	Нет поворота строк.

ENUM_SVDQRP_R

Перечисление, определяющее необходимость транспонирования треугольной матрицы R, полученной в результате начальной QR-факторизации.

Идентификатор	Описание
SVDQRPR_T	После первоначальной повернутой QR-факторизации GESVD применяется к транспонированному R**T вычисленного треугольного множителя R.
SVDQRPR_N	Треугольный множитель R задается как входные данные для GESVD.

ENUM_SVDQRP_U

Перечисление, определяющее способ вычисления левых сингулярных векторов.

Идентификатор	Описание
SVDQRPU_A	Вычисляются все M левых сингулярных векторов.
SVDQRPU_S	Вычисляется min(M,N) левых сингулярных векторов.
SVDQRPU_R	Определяется числовой ранг NUMRANK и вычисляется только NUMRANK левых сингулярных векторов.
SVDQRPU_F	Возвращаются N левых сингулярных векторов.
SVDQRPU_N	Левые сингулярные векторы не вычисляются.

ENUM_SVDQRP_V

© 2000-2025, MetaQuotes Ltd.

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
SVDQRPV_A	Все N правых сингулярных векторов вычисляются.
SVDQRPV_R	Числовой ранг NUMRANK определяется и вычисляются только NUMRANK правых сингулярных векторов.
SVDQRPV_N	Правые сингулярные векторы не вычисляются.

Смотри также

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionBisect

Singular Value Decomposition — разложение по сингулярным значениям, алгоритм bisection (lapack-функция GESVDX).

Вычисления для типа matrix<double>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,      // уровень точности расчёта
   ENUM_BLAS_RANGE      range,     // подмножество вычисляемых сингулярных значений
   double               lower,     // нижняя граница подмножества
   double               upper,     // верхняя граница подмножества
   vector&              S,         // вектор рассчитанных сингулярных значений
   matrix&              U,         // матрица рассчитанных левых векторов U
   matrix&              VT         // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,      // уровень точности расчёта
   ENUM_BLAS_RANGE      range,     // подмножество вычисляемых сингулярных значений
   double               lower,     // нижняя граница подмножества
   double               upper,     // верхняя граница подмножества
   vectorf&             S,         // вектор рассчитанных сингулярных значений
   matrixf&              U,         // матрица рассчитанных левых векторов U
   matrixf&             VT         // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,      // уровень точности расчёта
   ENUM_BLAS_RANGE      range,     // подмножество вычисляемых сингулярных значений
   double               lower,     // нижняя граница подмножества
   double               upper,     // верхняя граница подмножества
   vector&              S,         // вектор рассчитанных сингулярных значений
   matrixc&             U,         // матрица рассчитанных левых векторов U
   matrixc&             VT         // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,               // уровень точности расчёта
   ENUM_BLAS_RANGE      range,              // подмножество вычисляемых сингулярных зна
   double               lower,              // нижняя граница подмножества
   double               upper,              // верхняя граница подмножества
   vectorf&             singular_values,    // вектор рассчитанных сингулярных значений
   matrixcf&            u,                  // матрица рассчитанных левых векторов U
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixcf&           vt                  // транспонированная матрица правых векторо
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_SVD_VECTORS, определяющее способ вычисления левых и правых сингулярных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых сингулярных значений и векторов.

lower

[in] Нижняя граница подмножества сингулярных значений, задаётся в зависимости от значения параметра range.

upper

[in] Верхняя граница подмножества сингулярных значений, задаётся в зависимости от значения параметра range.

S

[out] Вектор сингулярных значений.

U

[out] Матрица левых сингулярных векторов.

VT

[out] Матрица правых сингулярных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значений параметров *jobuv* и *range*.

При значении BLASRANGE_A все сингулярные значения вычисляются, при этом параметры lower и upper игнорируются.

При значении BLASRANGE_V вычисляются только те сингулярные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper.

При значении BLASRANGE_I вычисляются только те сингулярные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три сингулярные значения.

ENUM_SVD_VECTORS

Перечисление, определяющее способ вычисления левых и правых сингулярных векторов.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
SVDVECTORS_N	Вычисляются только сингулярные значения, без векторов.
SVDVECTORS_U	Вычисляются левые сингулярные векторы.
SVDVECTORS_V	Вычисляются правые сингулярные векторы.
SVDVECTORS_UV	Вычисляются левые и правые сингулярные векторы.

ENUM_BLAS_RANGE

Перечисление, определяющее подмножество вычисляемых сингулярных значений и векторов.

Идентификатор	Описание
BLASRANGE_A	Будут найдены все сингулярные или собственные значения.
BLASRANGE_V	Будут найдены все сингулярные или собственные значения в полуоткрытом интервале (VL,VU].
BLASRANGE_I	Будут найдены сингулярные или собственные значения с IL-го по IU-е

Смотри также

SingularValueDecompositionDC, SingularValueDecompositionQR, SingularValueDecompositionQRPivot

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionJacobiHigh

Singular Value Decomposition — разложение по сингулярным значениям, алгоритм Jacobi high level (lapack-функция GEJSV).

Вычисления для типа matrix<double>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // уровень точности расчёта
   ENUM_SVDJH_U   jobu,            // способ вычисления левых векторов
   ENUM_SVDJH_V   jobv,            // способ вычисления правых векторов
   ENUM_SVDJH_R   jobr,            // определение диапазона вычисляемых сингулярных зн
   ENUM_SVDJH_T   jobt,            // определение необходимости транспонирования при в
   ENUM_SVDJH_P   jobp,            // определение возможности структурированных пертур
   vector&        S,               // вектор рассчитанных сингулярных значений
   matrix&        U,               // матрица рассчитанных левых векторов U
   matrix&        V,               // матрица рассчитанных левых векторов V
   vector&        work_results     // дополнительные результаты вычисления
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // уровень точности расчёта
   ENUM_SVDJH_U   jobu,            // способ вычисления левых векторов
   ENUM_SVDJH_V   jobv,            // способ вычисления правых векторов
   ENUM_SVDJH_R   jobr,            // определение диапазона вычисляемых сингулярных зн
   ENUM_SVDJH_T   jobt,            // определение необходимости транспонирования при в
   ENUM_SVDJH_P   jobp,            // определение возможности структурированных пертур
   vectorf&       S,               // вектор рассчитанных сингулярных значений
   matrixf&       U,               // матрица рассчитанных левых векторов U
   matrixf&       V,               // матрица рассчитанных левых векторов V
   vector&        work_results     // дополнительные результаты вычисления
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // уровень точности расчёта
   ENUM_SVDJH_U   jobu,            // способ вычисления левых векторов
   ENUM_SVDJH_V   jobv,            // способ вычисления правых векторов
   ENUM_SVDJH_R   jobr,            // определение диапазона вычисляемых сингулярных зн
   ENUM_SVDJH_T   jobt,            // определение необходимости транспонирования при в
   ENUM_SVDJH_P   jobp,            // определение возможности структурированных пертур
   vectorc&       S,               // вектор рассчитанных сингулярных значений
   matrixc&       U,               // матрица рассчитанных левых векторов U
   matrixc&       V,               // матрица рассчитанных левых векторов V
   vector&        work_results     // дополнительные результаты вычисления
   );
```

© 2000-2025, MetaQuotes Ltd.

Вычисления для типа matrix<complexf>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A    joba,                  // уровень точности расчёта
   ENUM_SVDJH_U    jobu,                  // способ вычисления левых векторов
   ENUM_SVDJH_V    jobv,                  // способ вычисления правых векторов
   ENUM_SVDJH_R    jobr,                  // определение диапазона вычисляемых сингул
   ENUM_SVDJH_T    jobt,                  // определение необходимости транспонирова
   ENUM_SVDJH_P    jobp,                  // определение возможности структурированны
   vectorf&        singular_values,       // вектор рассчитанных сингулярных значений
   matrixcf&       u,                     // матрица рассчитанных левых векторов U
   matrixcf&       v,                     // матрица рассчитанных левых векторов V
   vectorf&        work_results           // дополнительные результаты вычисления
   );
```

Параметры

joba

[in] Значение из перечисления ENUM_SVDJH_A, определяющее уровень точности расчёта SVD.

jobu

[in] Значение из перечисления ENUM_SVDJH_U, определяющее способ вычисления левых сингулярных векторов.

jobv

[in] Значение из перечисления ENUM_SVDJH_V, определяющее способ вычисления правых сингулярных векторов.

jobr

[in] Значение из перечисления ENUM_SVDJH_R, определяющее диапазон вычисляемых значений.

jobt

[in] Значение из перечисления ENUM_SVDJH_T, определяющее необходимость транспонирования матрицы в случае, когда он квадратная. Если матрица неквадратная, то этот параметр игнорируется.

jobp

[in] Значение из перечисления ENUM_SVDJH_P, определяющее возможность структурированных пертурбаций для удаления денормализованных значений.

S

[out] Вектор сингулярных значений.

При work(1)/work(2) = единица: сингулярные значения A. Во время вычисления S содержит евклидовы нормы столбцов итерационных матриц в массиве a.

Для work(1)≠work(2): сингулярные значения A равны (work(1)/work(2)) * S(1:n). Такая факторизованная форма используется в случаях, когда σ_max(A) выходит за пределы диапазона представления чисел (overflow, переполнение) или если малые сингулярные значения были сохранены от потери точности (underflow) путем масштабирования входной матрицы A.

© 2000-2025, MetaQuotes Ltd.

При jobr = 'R' некоторые сингулярные значения могут быть возвращены как точные нули, полученные путем "обнуления", если они ниже порога численного ранга или являются денормализованными числами.

U

[out] Матрица левых сингулярных векторов.

V

[out] Матрица правых сингулярных векторов.

work_results

[out] Вектор, состоящий из 7 статистик, полученных в результате расчёта.

work(1) = scale = work(2)/work(1) — коэффициент масштабирования, такой что scale*sva(1:n) — вычисленные сингулярные значения A. См. также описание S.

work(2) = work(1) — см. описание work(1).

work(3) = sconda — это оценка числа обусловленности столбца, уравновешенного A. Если joba = 'E' или 'G', sconda — это оценка sqrt(||(R**t * R)**(-1)||_1). Вычисляется с использованием ?pocon. Удовлетворяет неравенству n**(-1/4) * sconda ≤ ||R**(-1)||_2 ≤ n**(1/4) * sconda, где R — треугольный множитель из QR-разложения матрицы A. Если же R усечен и численный ранг оказывается строго меньше n, sconda возвращается как -1, что означает, что самые малые сингулярные значения могут быть утеряны.

Если требуется полное разложение по сингулярным значениям SVD, для анализа алгоритма полезны следующие два числа обусловленности. Для из использования нужно понимать особенности и детали метода.

work(4) — оценка числа обусловленности треугольного множителя в первом QR-разложении.

work(5) — оценка числа обусловленности треугольного множителя во втором QR-разложении.

Если jobt = 'T', вычисляются два дополнительных параметра. Для их использования необходимо понимать особенности и детали метода.

work(6) — энтропия матрицы A**t*A, равная энтропии Шеннона diag(A**t*A) / Trace(A**t*A), взятой как точка в симплексе вероятности.

work(7) — энтропия матрицы A*A**t.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Количество строк матрицы не должно быть меньше количества столбцов.

ENUM_SVDJH_A

Перечисление, определяющее уровень точности расчёта SVD.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
SVDJHA_C	Вычисление как в 'C', но с дополнительной оценкой числа обусловленности. Это дает более реалистичную границу погрешности.
SVDJHA_E	Вычисление как в SVDJHA_C, но с дополнительной оценкой числа обусловленности. Это дает более реалистичную границу погрешности.
SVDJHA_F	Более высокая точность, чем у SVDJHA_C.
SVDJHA_G	Вычисление как в SVDJHA_F, но с дополнительной оценкой числа обусловленности.
SVDJHA_A	Малые сингулярные значения считаются шумом, и матрица рассматривается как численно вырожденная по рангу.
SVDJHA_R	Аналогично SVDJHA_A, но с повышенной точностью.

ENUM_SVDJH_U

Перечисление, определяющее способ вычисления левых сингулярных векторов.

Идентификатор	Описание
SVDJHU_U	Возвращаются N столбцов матрицы U в массиве U.
SVDJHU_F	Возвращается полный набор из M левых сингулярных векторов в массиве U.
SVDJHU_N	Матрица U не вычисляется.

ENUM_SVDJH_V

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
SVDJHV_V	Возвращаются N столбцов матрицы V в массиве V.
SVDJHV_J	В массиве V возвращаются N столбцов матрицы V, но они вычисляются как произведение вращений Якоби.
SVDJHV_N	V is not computed

ENUM_SVDJH_R

Перечисление, определяющее диапазон вычисляемых значений.

Идентификатор	Описание
SVDJHR_N	Не обнулять малые столбцы матрицы c*A.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
SVDJHR_R	Ограниченный диапазон для sigma(c*A). Этот вариант рекомендуется.

ENUM_SVDJH_T

Перечисление, определяющее необходимость транспонирования матрицы в случае, когда он квадратная.

Идентификатор	Описание
SVDJHT_T	Транспонировать, если энтропийный тест указывает на возможную более быструю сходимость процесса Якоби.
SVDJHT_N	Не использовать транспонирование. Не делать предположений.

ENUM_SVDJH_P

Перечисление, определяющее возможность структурированных пертурбаций для удаления денормализованных значений.

Идентификатор	Описание
SVDJHP_P	Использовать пертурбацию.
SVDJHP_N	Не использовать пертурбацию. Do not perturb.

Смотри также

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionJacobiLow

Singular Value Decomposition, алгоритм Jacobi low level (lapack-функция GESVJ). В некоторых случаях вычисляет малые сингулярные значения и их сингулярные векторы гораздо более точно, чем другие подпрограммы SVD.

Вычисления для типа matrix<double>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // способ вычисления левых векторов
   ENUM_SVDJH_V   jobv,            // способ вычисления правых векторов
   double         ctol,            // порог сходимости (threshold for convergence), ec
   ulong          mv,              // количество первых строк матрицы V, если jobv='A'
   vector&        S,               // вектор рассчитанных сингулярных значений
   matrix&        U,               // матрица рассчитанных левых векторов U
   matrix&        V,               // матрица рассчитанных левых векторов V
   vector&        work_results     // дополнительные результаты вычисления
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // способ вычисления левых векторов
   ENUM_SVDJH_V   jobv,            // способ вычисления правых векторов
   double         ctol,            // порог сходимости (threshold for convergence), ec
   ulong          mv,              // количество первых строк матрицы V, если jobv='A'
   vectorf&       S,               // вектор рассчитанных сингулярных значений
   matrixf&       U,               // матрица рассчитанных левых векторов U
   matrixf&       V,               // матрица рассчитанных левых векторов V
   vector&        work_results     // дополнительные результаты вычисления
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // способ вычисления левых векторов
   ENUM_SVDJH_V   jobv,            // способ вычисления правых векторов
   double         ctol,            // порог сходимости (threshold for convergence), ec
   ulong          mv,              // количество первых строк матрицы V, если jobv='A'
   vector&        S,               // вектор рассчитанных сингулярных значений
   matrixc&       U,               // матрица рассчитанных левых векторов U
   matrixc&       V,               // матрица рассчитанных левых векторов V
   vector&        work_results     // дополнительные результаты вычисления
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U    jobu,                  // способ вычисления левых векторов
   ENUM_SVDJH_V    jobv,                  // способ вычисления правых векторов
```

© 2000-2025, MetaQuotes Ltd.

```
   double          ctol,                  // порог сходимости (threshold for converge
   ulong           mv,                    // количество первых строк матрицы V, если
   vectorf&        singular_values,       // вектор рассчитанных сингулярных значений
   matrixcf&       u,                     // матрица рассчитанных левых векторов U
   matrixcf&       v,                     // матрица рассчитанных левых векторов V
   vectorf&        work_results           // дополнительные результаты вычисления
   );
```

Параметры

jobu

[in] Значение из перечисления ENUM_SVDJL_U, определяющее способ вычисления левых сингулярных векторов.

jobv

[in] Значение из перечисления ENUM_SVDJL_V, определяющее способ вычисления правых сингулярных векторов.

ctol

[in] Порог сходимости, если jobu=SVDJLU_C. При других значениях jobu параметр игнорируется.

mv

[in] Количество рассчитываемых строк матрицы V, если jobv=SVDJLV_A. При других значениях jobv параметр игнорируется.

S

[out] Вектор сингулярных значений.

В зависимости от значения scale = work(1), где scale — коэффициент масштабирования:

Если scale = 1, S(1:n) содержит вычисленные сингулярные значения матрицы a. В процессе вычислений массив sva содержит евклидовы нормы столбцов итерационных матриц в массиве a.

Если scale ≠ 1, сингулярные значения матрицы a вычисляются как scale*S(1:n). Такая факторизованная форма используется, если некоторые сингулярные значения a могут переполниться (overflow) или потеряться из-за недостаточной точности (underflow).

U

[out] Матрица левых сингулярных векторов.

V

[out] Матрица правых сингулярных векторов (нетранспонированная).

work_results

[out] Вектор, состоящий из 7 статистик, полученных в результате расчёта.

work(1) = scale — коэффициент масштабирования, такой что scale*S(1:n) — вычисленные сингулярные значения A. См. также описание S.

work(2) — количество ненулевых вычисленных сингулярных значений.

work(3) — количество вычисленных сингулярных значений, превышающих порог потери точности (underflow).

© 2000-2025, MetaQuotes Ltd.

work(4) — число итераций вращений Якоби, необходимых для численной сходимости.

work(5) = max_{i.NE.j} |COS(A(:,i),A(:,j))| в последнем цикле. Полезно в случаях, когда ? gesvj не сошёлся, так как позволяет оценить, можно ли использовать результат, а также применимо для пост-фактум анализа.

work(6) — наибольшее абсолютное значение среди синусов углов вращений Якоби в последнем цикле. Может быть полезно для пост-фактум анализа.

Возвращаемое значение

Возвращает true в случае успеха или false, если произошла ошибка.

Примечание

Количество строк матрицы не должно быть меньше количества столбцов.

ENUM_SVDJL_U

Перечисление, определяющее способ вычисления левых сингулярных векторов.

Идентификатор	Описание
SVDJLU_U	Левые сингулярные векторы, соответствующие ненулевым сингулярным значениям, вычисляются и возвращаются в ведущих столбцах матрицы A
SVDJLU_C	Аналогично SVDJLU_U, но с возможностью управления уровнем численной ортогональности вычисленных левых сингулярных векторов.
SVDJLU_N	Матрица U не вычисляется.

ENUM_SVDJL_V

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
SVDJLV_V	Матрица V вычисляется
SVDJLV_A	Вращения Якоби применяются к матрице V размером MV на N.
SVDJLV_N	Матрица V не вычисляется.

Смотри также

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagDC

Singular Value Decomposition, алгоритм divide and conquer для бидиагональной матрицы (lapack-функция BDSDC).

Вычисления для типа matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagDC(
   ENUM_SVDBIDIAG_Z    jobz,       // способ вычисления левых векторов
   ENUM_BLAS_RANGE     range,      // подмножество вычисляемых сингулярных значений
   double              lower,      // нижняя граница подмножества
   double              upper,      // верхняя граница подмножества
   vector&             S,          // вектор рассчитанных сингулярных значений
   matrix&             U,          // матрица рассчитанных левых векторов U
   matrix&             VT          // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagDC(
   ENUM_SVDBIDIAG_Z    jobz,       // способ вычисления левых векторов
   ENUM_BLAS_RANGE     range,      // подмножество вычисляемых сингулярных значений
   double              lower,      // нижняя граница подмножества
   double              upper,      // верхняя граница подмножества
   vectorf&            S,          // вектор рассчитанных сингулярных значений
   matrixf&            U,          // матрица рассчитанных левых векторов U
   matrixf&            VT          // транспонированная матрица правых векторов VT
   );
```

Параметры

jobz

[in] Значение из перечисления ENUM_SVDBIDIAG_Z, определяющее способ вычисления левых сингулярных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых сингулярных значений и векторов.

lower

[in] Нижняя граница подмножества сингулярных значений, задаётся в зависимости от значения параметра range.

upper

[in] Верхняя граница подмножества сингулярных значений, задаётся в зависимости от значения параметра range.

S

[out] Вектор сингулярных значений.

U

[out] Матрица левых сингулярных векторов.

© 2000-2025, MetaQuotes Ltd.

V

[out] Транспонированная матрица правых сингулярных векторов.

Возвращаемое значение

Возвращает true в случае успеха или false, если произошла ошибка.

Примечание

Расчёт зависит от значений параметров jobz и и range.

При значении BLASRANGE_A все сингулярные значения вычисляются, при этом параметры lower и upper игнорируются.

При значении BLASRANGE_V вычисляются только те сингулярные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper

При значении BLASRANGE_I вычисляются только те сингулярные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три сингулярные значения.

Бидиагональная матрица — квадратная матрица с ненулевыми главной диагональю и одной из субдиагоналей.

Верхняя бидиагональная матрица

```
[[x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x],
 [0, 0, 0, 0, x]]
```

Нижняя бидиагональная матрица

```
[[x, 0, 0, 0, 0],
 [x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x]]
```

ENUM_SVDBIDIAG_Z

Перечисление, определяющее способ вычисления левых сингулярных векторов.

Идентификатор	Описание
SVDJOBZ_V	Compute singular values and singular vectors.
SVDJOBZ_N	Compute singular values only.

© 2000-2025, MetaQuotes Ltd.

ENUM_BLAS_RANGE

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
BLASRANGE_A	Вычисляются все сингулярные или собственные значения.
BLASRANGE_V	Вычисляются все сингулярные или собственные значения в полуоткрытом интервале (VL,VU).
BLASRANGE_I	Вычисляются особые или собственные значения с IL-го по IU-е.

Смотри также

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagBisect

Singular Value Decomposition, алгоритм bisection для бидиагональных матриц (lapack-функция BDSVDX).

Вычисления для типа matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagBisect(
   ENUM_SVDBIDIAG_Z    jobz,       // способ вычисления левых векторов
   ENUM_BLAS_RANGE     range,      // подмножество вычисляемых сингулярных значений
   double              lower,      // нижняя граница подмножества
   double              upper,      // верхняя граница подмножества
   vector&             S,          // вектор рассчитанных сингулярных значений
   matrix&             U,          // матрица рассчитанных левых векторов U
   matrix&             VT          // транспонированная матрица правых векторов VT
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagBisect(
   ENUM_SVDBIDIAG_Z    jobz,       // способ вычисления левых векторов
   ENUM_BLAS_RANGE     range,      // подмножество вычисляемых сингулярных значений
   double              lower,      // нижняя граница подмножества
   double              upper,      // верхняя граница подмножества
   vectorf&            S,          // вектор рассчитанных сингулярных значений
   matrixf&            U,          // матрица рассчитанных левых векторов U
   matrixf&            VT          // транспонированная матрица правых векторов VT
   );
```

Параметры

jobz

[in] Значение из перечисления ENUM_SVDBIDIAG_Z, определяющее способ вычисления левых сингулярных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых сингулярных значений и векторов.

lower

[in] Нижняя граница подмножества сингулярных значений, задаётся в зависимости от значения параметра range.

upper

[in] Верхняя граница подмножества сингулярных значений, задаётся в зависимости от значения параметра range.

S

[out] Вектор сингулярных значений.

U

[out] Матрица левых сингулярных векторов.

© 2000-2025, MetaQuotes Ltd.

V

[out] Транспонированная матрица правых сингулярных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

При значении BLASRANGE_A все сингулярные значения вычисляются, при этом параметры lower и upper игнорируются.

При значении BLASRANGE_V вычисляются только те сингулярные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper

При значении BLASRANGE_I вычисляются только те сингулярные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три сингулярные значения.

Бидиагональная матрица — квадратная матрица с ненулевыми главной диагональю и одной из субдиагоналей.

Верхняя бидиагональная матрица

```
[[x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x],
 [0, 0, 0, 0, x]]
```

Нижняя бидиагональная матрица

```
[[x, 0, 0, 0, 0],
 [x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x]]
```

ENUM_SVDBIDIAG_Z

Перечисление, определяющее способ вычисления левых сингулярных векторов.

Идентификатор	Описание
SVDJOBZ_V	Вычисляются сингулярные значения и сингулярные векторы.
SVDJOBZ_N	Вычисляются только сингулярные значения.

ENUM_BLAS_RANGE

© 2000-2025, MetaQuotes Ltd.

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
BLASRANGE_A	Вычисляются все сингулярные или собственные значения.
BLASRANGE_V	Вычисляются все сингулярные или собственные значения в полуоткрытом интервале (VL,VU).
BLASRANGE_I	Вычисляются особые или собственные значения с IL-го по IU-е.

Смотри также

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000-2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagQR

Вычисление сингулярного разложения общей матрицы, предварительно приведенной к бидиагональному виду методом ReduceToBidiagonal. Lapack-функция BDSQR.

Вычисления для типа matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrix&          Q,          // ортогональная матрица Q
   matrix&          PT,         // транспонированная матрица P
   vector&          S,          // вектор вычисленных сингулярных значений
   matrix&          U,          // матрица U с вычисленными левыми векторами
   matrix&          VT          // транспонированная матрица V с правыми векторами
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrix&          Q,          // ортогональная матрица Q
   matrix&          PT,         // транспонированная матрица P
   vectorf&         S,          // вектор вычисленных сингулярных значений
   matrixf&         U,          // матрица U с вычисленными левыми векторами
   matrixf&         VT          // транспонированная матрица V с правыми векторами
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrix&          Q,          // ортогональная матрица Q
   matrix&          PH,         // эрмитово сопряженная матрица P
   vector           S,          // вектор вычисленных сингулярных значений
   matrixc          U,          // матрица U с вычисленными левыми векторами
   matrixc          VH          // эрмитово сопряженная матрица V с правыми векторами
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrixf&         Q,          // ортогональная матрица Q
   matrixf&         PH,         // эрмитово сопряженная матрица P
   vectorf&         S,          // вектор вычисленных сингулярных значений
   matrixcf&        U,          // матрица U с вычисленными левыми векторами
   matrixcf&        VH          // эрмитово сопряженная матрица V с правыми векторами
   );
```

Параметры

Q

[in] Ортогональная матрица Q, полученная методом ReflectBidiagonalToQP. Если матрица Q имеет нулевой размер, то левые сингулярные векторы U не вычисляются.

PT

© 2000-2025, MetaQuotes Ltd.

[in] Транспонированная (или эрмитово-сопряжённая — в случае комплексной матрицы) матрица P, полученная методом ReflectBidiagonalToQP. Если матрица PT имеет нулевой размер, то правые сингулярные векторы VT не вычисляются.

S

[out] Вектор сингулярных значений.

U

[out] Матрица левых сингулярных векторов.

VT

[out] Матрица правых сингулярных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

BDSQR вычисляет сингулярные значения и, при необходимости, правые и/или левые сингулярные векторы из сингулярного разложения (SVD) матрицы B размером N на N (верхней или нижней) с бидиагона́льной формой, используя неявный QR-алгоритм с нулевым сдвигом. Сингулярное разложение матрицы B имеет вид:

B = Q * S * P**T

где S — диагональная матрица сингулярных значений, Q — ортогональная матрица левых сингулярных векторов, а P — ортогональная матрица правых сингулярных векторов. Если нужно вычислить левые сингулярные векторы, эта вместо Q возвращается U□Q, а если правые сингулярные векторы, вместо PT возвращается P**T*VT для заданных вещественных входных матриц U и VT. Если U и VT — ортогональные матрицы, которые сводят общую матрицу A к бидиагона́льной форме: A = U□B□VT, как при вычислении с помощью GEBRD, тогда

A = (U*Q) * S * (P**T*VT)

— это сингулярное разложение матрицы A.

© 2000-2025, MetaQuotes Ltd.

Eigen Values

Раздел, содержащий функции для вычисления собственных значений и собственных векторов, предоставляет методы решения стандартных задач линейной алгебры с использованием алгоритмов библиотеки LAPACK. Эти функции полезны для анализа матриц, диагонализации, стабилизации систем и других задач.

- **EigenSolver**: Функция предназначена для вычисления собственных значений и собственных векторов произвольной квадратной матрицы с использованием классического алгоритма, представленного lapack-функцией GEEV. Этот метод применяется для широкого класса матриц и позволяет получать разложение матриц на их собственные значения и векторы.
- **EigenSymmetricDC**: Функция для вычисления собственных значений и собственных векторов симметричных или эрмитовых матриц, применяя алгоритм "разделяй и властвуй" (divide and conquer). Используются lapack-функции SYEVD и HEEVD, что позволяет эффективно работать с симметричными или эрмитовыми матрицами, обеспечивая более быструю и точную обработку таких матриц.

Функция	Действие
EigenSolver	Вычисление собственных значений и собственных векторов обычной квадратной матрицы классическим алгоритмом (lapack-функция GEEV).
EigenSolverX	Вычисление собственных значений и собственных векторов обычной квадратной матрицы в режиме эксперта (Expert mode), то есть с возможностью влиять на алгоритм расчёта и возможностью получить сопутствующие расчётные данные (lapack-функция GEEVX).
EigenSolverShur	Вычисление собственных значений, верхнетреугольной матрицы в форме Шура и матрицы векторов Шура (lapack-функция GEES). Смотри Разложение Шура.
EigenSolver2	Вычисление обобщённых собственных значений и собственных векторов для пары обычных квадратных матриц (lapack-функция GGEV).
EigenSolver2X	Вычисление обобщённых собственных значений и собственных векторов для пары обычных квадратных матриц в режиме эксперта (Expert mode), то есть с возможностью влиять на алгоритм расчёта и возможностью получить сопутствующие расчётные данные (lapack-функция GGEVX). Обе матрицы должны быть одинакового размера.
EigenSolver2Shur	Вычисление для пары обычных квадратных матриц обобщённых собственных значений, обобщённых собственных векторов, обобщённых форм Шура и левых и правых векторов Шура (lapack-функция GGES).

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
EigenSolver2Blocked	Вычисление обобщённых собственных значений и собственных векторов для пары обычных квадратных матриц при помощи блочного алгоритма (lapack-функция GGEV3). Обе матрицы должны быть одинакового размера. Параметры метода точно такие же, как и у EigenSolver2.
EigenSolver2ShurBlocked	Вычисление для пары обычных квадратных матриц обобщённых собственных значений, обобщённых собственных векторов, обобщённых форм Шура и левых и правых векторов Шура при помощи блочного алгоритма (lapack-функция GGES3).
EigenSymmetricDC	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма "divide and conquere" (lapack-функции SYEVD, HEEVD).
EigenSymmetricQR	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи классического алгоритма QR (lapack-функции SYEV, HEEV).
EigenSymmetricRobust	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма Multiple Relatively Robust Representations, MRRR (lapack-функции SYEVR, HEEVR).
EigenSymmetricBisect	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма bisection (lapack-функции SYEVX, HEEVX).
EigenSymmetricDC	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма "divide and conquere" (lapack-функции SYEVD, HEEVD).
EigenSymmetricQR	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи классического алгоритма QR (lapack-функции SYEV, HEEV).
EigenSymmetricRobust	Вычисление собственных значений и собственных векторов симметричной или эрмитовой

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
	(комплексно-сопряжённой) матрицы при помощи алгоритма Multiple Relatively Robust Representations, MRRR (lapack-функции SYEVR, HEEVR).
EigenSymmetricBisect	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма bisection (lapack-функции SYEVX, HEEVX).
EigenSymmetricDC2s	Вычисление всех собственных значений и, при необходимости, собственных векторов вещественной симметричной или эрмитовой (комплексно-сопряжённой) матрицы с использованием двухэтапного алгоритма приведения к тридиагональному виду. Если требуется вычисление собственные векторы, применяется алгоритм divide and conquer («разделяй и властвуй») (функции lapack SYEVD_2STAGE, HEEVD_2STAGE).
EigenSymmetricQR2s	Вычисление всех собственных значений и, при необходимости, собственных векторов вещественной симметричной или эрмитовой (комплексно-сопряжённой) матрицы с использованием двухэтапного алгоритма приведения к тридиагональному виду (функции lapack SYEV_2STAGE, HEEV_2STAGE).
EigenSymmetricRobust2s	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи двухэтапного алгоритма: сначала приведение к тридиагональному виду, затем применение алгоритма множественных относительно надежных представлений (Multiple Relatively Robust Representations, MRRR) (функции lapack SYEVR_2STAGE, HEEVR_2STAGE).
EigenSymmetricBisect2s	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи двухэтапного алгоритма: сначала приведение к тридиагональному виду, затем применение метода бисекции (функции lapack SYEVX_2STAGE, HEEVX_2STAGE).
EigenSymmetric2DC	Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
	вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.
EigenSymmetric2QR	Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.
EigenSymmetric2Bisect	Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.
EigenTridiagonalDC	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием алгоритма "divide and conquer" (lapack-функция STEVD).
EigenTridiagonalQR	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием QR-алгоритма (lapack-функция STEV).
EigenTridiagonalRobust	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы при помощи алгоритма Multiple Relatively Robust Representations, MRRR (lapack-функция STEVR).
EigenTridiagonalBisect	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием алгоритма бисекции (lapack-функция STEVX).
EigenTridiagonalQL	Вычисление всех собственных значений симметричной трехдиагональной матрицы с использованием варианта QR-алгоритма без квадратного корня (lapack-функция STERF).
EigenTridiagonalDCQ	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием алгоритма "разделяй и властвуй" (lapack-функция STEDC).
EigenTridiagonalQRQ	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием QR-алгоритма (lapack-функция STEQR).

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
EigenTridiagonalPosDefQ	Вычисление собственных значений и собственных векторов симметричной положительно определенной трехдиагональной матрицы с использованием QR-алгоритма (lapack-функция PTEQR).

© 2000-2025, MetaQuotes Ltd.

General Matrices

Функции вычисления собственных значений и собственных векторов квадратной матрицы с использованием классических алгоритмов. В нем предоставляются различные методы для работы как с действительными, так и с комплексными матрицами, позволяя решать задачи линейной алгебры с выбором способов вычисления собственных векторов.

Функция	Действие
EigenSolver	Вычисление собственных значений и собственных векторов обычной квадратной матрицы классическим алгоритмом (lapack-функция GEEV).
EigenSolverX	Вычисление собственных значений и собственных векторов обычной квадратной матрицы в режиме эксперта (Expert mode), то есть с возможностью влиять на алгоритм расчёта и возможностью получить сопутствующие расчётные данные (lapack-функция GEEVX).
EigenSolverShur	Вычисление собственных значений, верхнетреугольной матрицы в форме Шура и матрицы векторов Шура (lapack-функция GEES). Смотри Разложение Шура.
EigenSolver2	Вычисление обобщённых собственных значений и собственных векторов для пары обычных квадратных матриц (lapack-функция GGEV).
EigenSolver2X	Вычисление обобщённых собственных значений и собственных векторов для пары обычных квадратных матриц в режиме эксперта (Expert mode), то есть с возможностью влиять на алгоритм расчёта и возможностью получить сопутствующие расчётные данные (lapack-функция GGEVX). Обе матрицы должны быть одинакового размера.
EigenSolver2Shur	Вычисление для пары обычных квадратных матриц обобщённых собственных значений, обобщённых собственных векторов, обобщённых форм Шура и левых и правых векторов Шура (lapack-функция GGES).
EigenSolver2Blocked	Вычисление обобщённых собственных значений и собственных векторов для пары обычных квадратных матриц при помощи блочного алгоритма (lapack-функция GGEV3). Обе матрицы должны быть одинакового размера. Параметры метода точно такие же, как и у EigenSolver2.
EigenSolver2ShurBlocked	Вычисление для пары обычных квадратных матриц обобщённых собственных значений, обобщённых

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
	собственных векторов, обобщённых форм Шура и левых и правых векторов Шура при помощи блочного алгоритма (lapack-функция GGES3).
EigenSymmetricDC	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма "divide and conquere" (lapack-функции SYEVD, HEEVD).
EigenSymmetricQR	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи классического алгоритма QR (lapack-функции SYEV, HEEV).
EigenSymmetricRobust	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма Multiple Relatively Robust Representations, MRRR (lapack-функции SYEVR, HEEVR).
EigenSymmetricBisect	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма bisection (lapack-функции SYEVX, HEEVX).
EigenSymmetricDC2s	Вычисление всех собственных значений и, при необходимости, собственных векторов вещественной симметричной или эрмитовой (комплексно-сопряжённой) матрицы с использованием двухэтапного алгоритма приведения к тридиагональному виду. Если требуется вычисление собственных векторов, применяется алгоритм divide and conquer («разделяй и властвуй») (функции lapack SYEVD_2STAGE, HEEVD_2STAGE).
EigenSymmetricQR2s	Вычисление всех собственных значений и, при необходимости, собственных векторов вещественной симметричной или эрмитовой (комплексно-сопряжённой) матрицы с использованием двухэтапного алгоритма приведения к тридиагональному виду (функции lapack SYEV_2STAGE, HEEV_2STAGE).
EigenSymmetricRobust2s	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
	двухэтапного алгоритма: сначала приведение к тридиагональному виду, затем применение алгоритма множественных относительно надежных представлений (Multiple Relatively Robust Representations, MRRR) (функции lapack SYEVR_2STAGE, HEEVR_2STAGE).
EigenSymmetricBisect2s	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи двухэтапного алгоритма: сначала приведение к тридиагональному виду, затем применение метода бисекции (функции lapack SYEVX_2STAGE, HEEVX_2STAGE).
EigenSymmetric2DC	Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.
EigenSymmetric2QR	Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.
EigenSymmetric2Bisect	Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.

© 2000-2025, MetaQuotes Ltd.

EigenSolver

Вычисление собственных значений и собственных векторов обычной квадратной матрицы классическим алгоритмом (LAPACK-функция GEEV).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // вычисление левых и правых собстве
   vectorc&              eigen_values,         // вектор рассчитанных собственных з
   matrix&               left_eigenvectors,    // матрица рассчитанных левых вектор
   matrix&               right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // вычисление левых и правых собстве
   vectorcf&             eigen_values,         // вектор рассчитанных собственных з
   matrixf&              left_eigenvectors,    // матрица рассчитанных левых вектор
   matrixf&              right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // вычисление левых и правых собстве
   vectorc&              eigen_values,         // вектор рассчитанных собственных з
   matrixc&              left_eigenvectors,    // матрица рассчитанных левых вектор
   matrixc&              right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // вычисление левых и правых собстве
   vectorcf&             eigen_values,         // вектор рассчитанных собственных з
   matrixcf&             left_eigenvectors,    // матрица рассчитанных левых вектор
   matrixcf&             right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VECTORS, определяющее способ вычисления левых и правых собственных векторов.

EV

[out] Вектор собственных значений.

left_eigenvectors

[out] Матрица левых собственных векторов.

© 2000-2025, MetaQuotes Ltd.

right_eigenvectors

[out] Матрица правых собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха и false в случае ошибки.

Примечание

Расчет зависит от значений параметра jobv.

Если параметр установлен в значение EIGVECTORS_N, левые и правые векторы не рассчитываются. Рассчитываются только собственные значения.

Если установлено EIGVECTORS_L, рассчитываются только левые собственные векторы, правые векторы не рассчитываются.

При значении EIGVECTORS_R рассчитываются только правые собственные векторы, левые векторы не рассчитываются.

При значении EIGVECTORS_LR рассчитываются левые и правые собственные векторы, собственные значения вычисляются всегда.

Реальные (не комплексные) матрицы могут иметь комплексное решение. Поэтому вектор собственных значений должен быть комплексным. В случае комплексного решения код ошибки устанавливается в 4019 (ERR_MATH_OVERFLOW). В противном случае нужно учитывать только реальные части комплексных значений вектора собственных значений.

ENUM_EIG_VECTORS

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVECTORS_N	Вычисляются только собственные значения, без векторов.
EIGVECTORS_L	Вычисляются только левые собственные векторы.
EIGVECTORS_R	Вычисляются только правые собственные векторы.
EIGVECTORS_LR	Вычисляются левые и правые собственные векторы, собственные значения вычисляются всегда.

© 2000-2025, MetaQuotes Ltd.

EigenSolverX

Вычисление собственных значений и собственных векторов обычной квадратной матрицы в режиме эксперта (Expert mode), то есть с возможностью влиять на алгоритм расчета и возможностью получить сопутствующие расчетные данные (Функция LAPACK GEEVX).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSolverX(
   ENUM_EIG_BALANCE      balance,              // способ балансировки входной матри
   ENUM_EIG_VECTORS      jobv,                 // необходимость вычисления левых и
   ENUM_EIG_SENSE        sense,                // необходимость вычисления обратных
   vectorc&              eigen_values,         // вектор рассчитанных собственных з
   matrix&               left_eigenvectors,    // матрица рассчитанных левых вектор
   matrix&               right_eigenvectors    // матрица рассчитанных правых векто
   matrix&               shur_matrix,          // сбалансированная матрица в форме
   long&                 ilo,                  // нижний индекс сбалансированной ма
   long&                 ihi,                  // верхний индекс сбалансированной м
   vector&               scale,                // подробности перестановок и масшта
   double&               ab_norm,              // 1-норма сбалансированной матрицы
   vector&               rconde,               // вектор обратных чисел обусловлен
   vector&               rcondv                // вектор обратных чисел обусловлен
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSolverX(
   ENUM_EIG_BALANCE      balance,              // способ балансировки входной матри
   ENUM_EIG_VECTORS      jobv,                 // необходимость вычисления левых и
   ENUM_EIG_SENSE        sense,                // необходимость вычисления обратных
   vectorcf&             eigen_values,         // вектор рассчитанных собственных з
   matrixf&              left_eigenvectors,    // матрица рассчитанных левых вектор
   matrixf&              right_eigenvectors    // матрица рассчитанных правых векто
   matrixf&              shur_matrix,          // сбалансированная матрица в форме
   long&                 ilo,                  // нижний индекс сбалансированной ма
   long&                 ihi,                  // верхний индекс сбалансированной м
   vectorf&              scale,                // подробности перестановок и масшта
   float&                ab_norm,              // 1-норма сбалансированной матрицы
   vectorf&              rconde,               // вектор обратных чисел обусловлен
   vectorf&              rcondv                // вектор обратных чисел обусловлен
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSolverX(
   ENUM_EIG_BALANCE      balance,              // способ балансировки входной матри
   ENUM_EIG_VECTORS      jobv,                 // необходимость вычисления левых и
   ENUM_EIG_SENSE        sense,                // необходимость вычисления обратных
   vectorc&              eigen_values,         // вектор рассчитанных собственных з
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixc&              left_eigenvectors,       // матрица рассчитанных левых вектор
   matrixc&              right_eigenvectors       // матрица рассчитанных правых векто
   matrixc&              shur_matrix,             // сбалансированная матрица в форме
   long&                 ilo,                     // нижний индекс сбалансированной ма
   long&                 ihi,                     // верхний индекс сбалансированной м
   vector&               scale,                   // подробности перестановок и масшта
   double&               ab_norm,                 // 1-норма сбалансированной матрицы
   vector&               rconde,                  // вектор обратных чисел обусловлен
   vector&               rcondv                   // вектор обратных чисел обусловлен
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSolverX(
   ENUM_EIG_BALANCE      balance,                 // способ балансировки входной матри
   ENUM_EIG_VECTORS      jobv,                    // необходимость вычисления левых и
   ENUM_EIG_SENSE        sense,                   // необходимость вычисления обратных
   vectorcf&             eigen_values,            // вектор рассчитанных собственных з
   matrixcf&             left_eigenvectors,       // матрица рассчитанных левых вектор
   matrixcf&             right_eigenvectors       // матрица рассчитанных правых векто
   matrixcf&             shur_matrix,             // сбалансированная матрица в форме
   long&                 ilo,                     // нижний индекс сбалансированной ма
   long&                 ihi,                     // верхний индекс сбалансированной м
   vectorf&              scale,                   // подробности перестановок и масшта
   float&                ab_norm,                 // 1-норма сбалансированной матрицы
   vectorf&              rconde,                  // вектор обратных чисел обусловлен
   vectorf&              rcondv                   // вектор обратных чисел обусловлен
   );
```

Параметры

balance

[in] Значение из перечисления ENUM_EIG_BALANCE, определяющее необходимость и способ балансировки входной матрицы; применяется для улучшения обусловленности собственных значений.

jobv

[in] Значение из перечисления ENUM_EIG_VECTORS, определяющее способ вычисления левых и правых собственных векторов.

sense

[in] Значение из перечисления ENUM_EIG_SENSE, определяющее необходимость вычисления обратных чисел обусловленности.

eigen_values

[out] Вектор собственных значений.

left_eigenvectors

[out] Матрица левых собственных векторов.

right_eigenvectors

© 2000-2025, MetaQuotes Ltd.

[out] Матрица правых собственных векторов.

shur_matrix

[out] Сбалансированная матрица в форме Шура. Не заполняется, если не рассчитываются ни левые, ни правые собственные векторы.

ilo

[out] Нижний индекс сбалансированной матрицы. Не заполняется, если балансировка не применяется.

ihi

[out] Верхний индекс сбалансированной матрицы. Не заполняется, если балансировка не применяется.

scale

[out] Вектор подробностей перестановок и масштабирования при балансировке входной матрицы.

Детали перестановок и масштабирования, применяемых при балансировке A.

Если P(j) — это индекс строки и столбца, меняемого местами со строкой и столбцом j, а D(j) — это коэффициент масштабирования, примененный к строке и столбцу j, тогда

scale(j) = P(j), для j = 1,...,ilo-1

= D(j), для j = ilo,...,ihi

= P(j) для j = ihi+1,..., n.

Порядок, в котором выполняются перестановки: сначала n в ihi+1, затем 1 в ilo-1.

ab_norm

[out] 1-норма сбалансированной матрицы (максимум из сумм абсолютных значений элементов любой из колонок матрицы).

rconde

[out] Вектор обратных чисел обусловленности для каждого собственного значения. Вычисляется, если значение параметра sense равно 'E' или 'B'.

rcondv

[out] Вектор обратных чисел обусловленности для каждого правого собственного вектора. Вычисляется, если значение параметра sense равно 'V' или 'B.

Возвращаемое значение

Возвращает true в случае успеха и false в случае ошибки.

Примечание

Балансировка входной матрицы зависит от значения параметра *balance*.

ENUM_EIG_BALANCE

© 2000-2025, MetaQuotes Ltd.

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGBALANCE_N	Не выполняется диагональное масштабирование и перестановка.
EIGBALANCE_P	Выполняются перестановки, чтобы привести матрицу к почти верхнетреугольному виду. Не выполняется диагональное масштабирование
EIGBALANCE_S	Выполняется диагональное масштабирование матрицы. Перестановка не проводится
EIGBALANCE_B	Выполняется и диагональное масштабирование, и перестановка.

ENUM_EIG_VECTORS

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVECTORS_N	Вычисляются только собственные значения, без векторов.
EIGVECTORS_L	Вычисляются только левые собственные векторы.
EIGVECTORS_R	Вычисляются только правые собственные векторы.
EIGVECTORS_LR	Вычисляются левые и правые собственные векторы, собственные значения вычисляются всегда.

ENUM_EIG_SENSE

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGSENSE_N	Не вычисляется ни одно из обратных чисел обусловленности.
EIGSENSE_E	Вычисляются только для собственных значений
EIGSENSE_V	Вычисляются только для правых собственных векторов
EIGSENSE_B	Вычисляется и для собственных значений, и для правых собственных векторов

© 2000-2025, MetaQuotes Ltd.

EigenSolverShur

Вычисление собственных значений, верхнетреугольной матрицы в форме Шура и матрицы векторов Шура (LAPACK-функция GEES). См. также Разложение Шура.

Вычисления для типа matrix<double>

```
bool  matrix::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // способ вычисления векторов Шура
   vectorc&              eigen_values,           // вектор рассчитанных собственных з
   matrix&               shur_matrix,            // матрица в форме Шура
   matrix&               shur_vectors            // матрица векторов Шура
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // способ вычисления векторов Шура
   vectorcf&             eigen_values,           // вектор рассчитанных собственных з
   matrixf&              shur_matrix,            // матрица в форме Шура
   matrixf&              shur_vectors            // матрица векторов Шура
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // способ вычисления векторов Шура
   vectorc&              eigen_values,           // вектор рассчитанных собственных з
   matrixc&              shur_matrix,            // матрица в форме Шура
   matrixc&              shur_vectors            // матрица векторов Шура
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSolverShur(
   ENUM_EIG_VECTORS      jobvs,                  // способ вычисления векторов Шура
   vectorcf&             eigen_values,           // вектор рассчитанных собственных з
   matrixcf&             shur_matrix,            // матрица в форме Шура
   matrixcf&             shur_vectors            // матрица векторов Шура
   );
```

Параметры

jobvs

[in] Значение из перечисления ENUM_EIG_SHUR, определяющее способ вычисления векторов Шура.

eigen_values

[out] Вектор собственных значений.

shur_matrix

[out] Верхнетреугольная матрица Шура (форма Шура для входной матрицы).

© 2000-2025, MetaQuotes Ltd.

shur_vectors

[out] Матрица векторов Шура. Не вычисляется, если параметр *jobvs* установлен в значение N.

Возвращаемое значение

Возвращает true в случае успеха и false в случае ошибки.

Примечание

Расчёт зависит от значений параметра *jobvs*.

Реальные (не комплексные) матрицы могут иметь комплексное решение. Поэтому входной вектор собственных значений должен быть комплексным. В случае комплексного решения код ошибки устанавливается в 4019 (ERR_MATH_OVERFLOW). В противном случае нужно учитывать только реальные части комплексных значений вектора собственных значений.

EigenSolverShur

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGSHUR_N	Векторы Шура не вычисляются
EIGSHUR_V	Векторы Шура вычисляются

© 2000-2025, MetaQuotes Ltd.

EigenSolver2

Вычисление обобщенных собственных значений и собственных векторов для пары обычных квадратных матриц (LAPACK-функция GGEV). Обе матрицы должны быть одинакового размера.

Вычисления для типа matrix<double>

```
bool  matrix::EigenSolver2(
   matrix&               B,                    // вторая матрица в паре
   ENUM_EIG_VECTORS      jobv,                 // способ расчета левых и правых век
   vectorc&              alpha,                // вектор рассчитанных собственных з
   vector&               beta,                 // вектор делителей собственных знач
   matrix&               left_eigenvectors,    // матрица рассчитанных левых вектор
   matrix&               right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSolver2(
   matrix&               B,                    // вторая матрица в паре
   ENUM_EIG_VECTORS      jobv,                 // способ расчета левых и правых век
   vectorcf&             alpha,                // вектор рассчитанных собственных з
   vectorf&              beta,                 // вектор делителей собственных знач
   matrixf&              left_eigenvectors,    // матрица рассчитанных левых вектор
   matrixf&              right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSolver2(
   matrixc&              B,                    // вторая матрица в паре
   ENUM_EIG_VECTORS      jobv,                 // способ расчета левых и правых век
   vectorc&              alpha,                // вектор рассчитанных собственных з
   vectorc&              beta,                 // вектор делителей собственных знач
   matrixc&              left_eigenvectors,    // матрица рассчитанных левых вектор
   matrixc&              right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSolver2(
   matrixcf&             B,                    // вторая матрица в паре
   ENUM_EIG_VECTORS      jobv,                 // способ расчета левых и правых век
   vectorcf&             alpha,                // вектор рассчитанных собственных з
   vectorcf&             beta,                 // вектор делителей собственных знач
   matrixcf&             left_eigenvectors,    // матрица рассчитанных левых вектор
   matrixcf&             right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Параметры

© 2000-2025, MetaQuotes Ltd.

B

[in] Вторая матрица в паре.

jobv

[in] Значение из перечисления ENUM_EIG_VECTORS, определяющее способ вычисления левых и правых собственных векторов.

alpha

[out] Вектор собственных значений.

beta

[out] Вектор делителей собственных значений.

left_eigenvectors

[out] Матрица левых собственных векторов.

right_eigenvectors

[out] Матрица правых собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха и false в случае ошибки.

Примечание

Расчет зависит от значений параметра jobv.

ENUM_EIG_VECTORS

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVECTORS_N	Вычисляются только собственные значения, без векторов.
EIGVECTORS_L	Вычисляются только левые собственные векторы.
EIGVECTORS_R	Вычисляются только правые собственные векторы.
EIGVECTORS_LR	Вычисляются левые и правые собственные векторы, собственные значения вычисляются всегда.

Обобщенное собственное значение для пары матриц (A,B) — это скалярное λ или такое отношение alpha/beta = lambda, при котором A - lambda*B является сингулярным. Обычно представляется в виде пары (alpha,beta), поскольку существует интерпретация для beta=0 и даже для alpha=0 и beta=0.

Правый собственный вектор v(j), соответствующий собственному значению lambda(j) матрицы (A,B), удовлетворяет условию

A * v(j) = lambda(j) * B * v(j).

© 2000-2025, MetaQuotes Ltd.

Левый собственный вектор u(j), соответствующий собственному значению lambda(j) матрицы (A,B), удовлетворяет условию

u(j)**H * A = lambda(j) * u(j)**H * B .

где u(j)**H — сопряженно-транспонированное значение u(j).

Реальные (не комплексные) матрицы могут иметь комплексное решение. Поэтому входной вектор собственных значений должен быть комплексным. В случае комплексного решения код ошибки устанавливается в 4019 (ERR_MATH_OVERFLOW). В противном случае нужно учитывать только реальные части комплексных значений вектора собственных значений.

© 2000-2025, MetaQuotes Ltd.

EigenSolver2X

Вычисление обобщённых собственных значений и собственных векторов для пары обычных квадратных матриц в режиме эксперта (Expert mode), то есть с возможностью влиять на алгоритм расчёта и возможностью получить сопутствующие расчётные данные (lapack-функция GGEVX). Обе матрицы должны быть одинакового размера.

Также может вычислять балансировочное преобразование для улучшения обусловленности собственных значений и векторов (ILO, IHI, LSCALE, RSCALE, ABNRM и BBNRM), обратных чисел обусловленности для собственных значений (RCONDE) и обратных чисел обусловленности для правых собственных векторов (RCONDV).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSolver2X(
   matrix&               B,                    // вторая матрица в паре
   ENUM_EIG_BALANCE      balance,              // способ балансировки входной матри
   ENUM_EIG_VECTORS      jobv,                 // необходимость вычисления левых и
   ENUM_EIG_SENSE        sense,                // необходимость вычисления обратных
   vectorc&              alpha,                // вектор рассчитанных собственных з
   vector&               beta,                 // вектор делителей собственных знач
   matrix&               left_eigenvectors,    // матрица рассчитанных левых вектор
   matrix&               right_eigenvectors    // матрица рассчитанных правых векто
   matrix&               shur_matrix1,         // первая часть вещественной формы Ш
   matrix&               shur_matrix2,         // вторая часть вещественной формы Ш
   long&                 ilo,                  // нижний индекс сбалансированной ма
   long&                 ihi,                  // верхний индекс сбалансированной м
   vector&               lscale,               // детали перестановок и масштабиров
   vector&               rscale,               // детали перестановок и масштабиров
   double&               ab_norm,              // 1-норма сбалансированной входной
   double&               bb_norm,              // 1-норма сбалансированной второй м
   vector&               rconde,               // вектор обратных чисел обусловлен
   vector&               rcondv                // вектор обратных чисел обусловлен
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSolver2X(
   matrixf&              B,                    // вторая матрица в паре
   ENUM_EIG_BALANCE      balance,              // способ балансировки входной матри
   ENUM_EIG_VECTORS      jobv,                 // необходимость вычисления левых и
   ENUM_EIG_SENSE        sense,                // необходимость вычисления обратных
   vectorcf&             alpha,                // вектор рассчитанных собственных з
   vectorf&              beta,                 // вектор делителей собственных знач
   matrixf&              left_eigenvectors,    // матрица рассчитанных левых вектор
   matrixf&              right_eigenvectors    // матрица рассчитанных правых векто
   matrixf&              shur_matrix1,         // первая часть вещественной формы Ш
   matrixf&              shur_matrix2,         // вторая часть вещественной формы Ш
   long&                 ilo,                  // нижний индекс сбалансированной ма
```

© 2000-2025, MetaQuotes Ltd.

```
   long&                ihi,                // верхний индекс сбалансированной м
   vectorf&             lscale,             // детали перестановок и масштабиро
   vectorf&             rscale,             // детали перестановок и масштабиро
   float&               ab_norm,            // 1-норма сбалансированной входной
   float&               bb_norm,            // 1-норма сбалансированной второй м
   vectorf&             rconde,             // вектор обратных чисел обусловлен
   vectorf&             rcondv              // вектор обратных чисел обусловлен
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSolver2X(
   matrixc&             B,                  // вторая матрица в паре
   ENUM_EIG_BALANCE     balance,            // способ балансировки входной матри
   ENUM_EIG_VECTORS     jobv,               // необходимость вычисления левых и
   ENUM_EIG_SENSE       sense,              // необходимость вычисления обратных
   vectorc&             alpha,              // вектор рассчитанных собственных з
   vectorc&             beta,               // вектор делителей собственных знач
   matrixc&             left_eigenvectors,  // матрица рассчитанных левых вектор
   matrixc&             right_eigenvectors  // матрица рассчитанных правых векто
   matrixc&             shur_matrix1,       // первая часть вещественной формы Ш
   matrixc&             shur_matrix2,       // вторая часть вещественной формы Ш
   long&                ilo,                // нижний индекс сбалансированной ма
   long&                ihi,                // верхний индекс сбалансированной м
   vector&              lscale,             // детали перестановок и масштабиро
   vector&              rscale,             // детали перестановок и масштабиро
   double&              ab_norm,            // 1-норма сбалансированной входной
   double&              bb_norm,            // 1-норма сбалансированной второй м
   vector&              rconde,             // вектор обратных чисел обусловлен
   vector&              rcondv              // вектор обратных чисел обусловлен
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSolver2X(
   matrixcf&            B,                  // вторая матрица в паре
   ENUM_EIG_BALANCE     balance,            // способ балансировки входной матри
   ENUM_EIG_VECTORS     jobv,               // необходимость вычисления левых и
   ENUM_EIG_SENSE       sense,              // необходимость вычисления обратных
   vectorcf&            alpha,              // вектор рассчитанных собственных з
   vectorf&             beta,               // вектор делителей собственных знач
   matrixcf&            left_eigenvectors,  // матрица рассчитанных левых вектор
   matrixcf&            right_eigenvectors  // матрица рассчитанных правых векто
   matrixcf&            shur_matrix1,       // первая часть вещественной формы Ш
   matrixcf&            shur_matrix2,       // вторая часть вещественной формы Ш
   long&                ilo,                // нижний индекс сбалансированной ма
   long&                ihi,                // верхний индекс сбалансированной м
   vectorf&             lscale,             // детали перестановок и масштабиро
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf&              rscale,                 // детали перестановок и масштабиро
   float&                ab_norm,                // 1-норма сбалансированной входной
   float&                bb_norm,                // 1-норма сбалансированной второй м
   vectorf&              rconde,                 // вектор обратных чисел обусловлен
   vectorf&              rcondv                  // вектор обратных чисел обусловлен
   );
```

Параметры

B

[in] Вторая матрица в паре.

balance

[in] Значение из перечисления ENUM_EIG_BALANCE, определяющее необходимость и способ балансировки входной матрицы; применяется для улучшения обусловленности собственных значений.

jobv

[in] Значение из перечисления ENUM_EIG_VECTORS, определяющее способ вычисления левых и правых собственных векторов.

sense

[in] Значение из перечисления ENUM_EIG_SENSE, определяющее необходимость вычисления обратных чисел обусловленности.

eigen_values

[out] Вектор собственных значений.

left_eigenvectors

[out] Матрица левых собственных векторов.

right_eigenvectors

[out] Матрица правых собственных векторов.

shur_matrix1, shur_matrix2

[out] 2 части сбалансированной матрицы в форме Шура; не заполняется, если не рассчитываются ни левые, ни правые собственные векторы.

ilo

[out] Нижний индекс сбалансированной матрицы; не заполняется, если балансировка не применяется.

ihi

[out] Верхний индекс сбалансированной матрицы; не заполняется, если балансировка не применяется.

lscale

[out] Вектор содержит детали перестановок и масштабирования, применяемых к левой стороне A и B.

Если PL(j) — это индекс строки, меняемой местами со строкой j, а DL(j) — это коэффициент масштабирования, примененный к строке j, тогда

lscale(j) = PL(j), для j = 1,..., ilo-1

© 2000-2025, MetaQuotes Ltd.

= DL(j), для j = ilo,...,ihi

= PL(j) для j = ihi+1,..., n.

Порядок, в котором выполняются перестановки: сначала n в ihi+1, затем 1 в ilo-1.

rscale

[out] Вектор содержит детали перестановок и масштабирования, применяемых к правой стороне A и B.

Если PR(j) — это индекс столбца, меняемого местами со столбцом j, а DR(j) — это коэффициент масштабирования, примененный к столбцу j, тогда

rscale(j) = PR(j), для j = 1,..., ilo-1

= DR(j), для j = ilo,...,ihi

= PR(j) для j = ihi+1,..., n.

Порядок, в котором выполняются перестановки: сначала n в ihi+1, затем 1 в ilo-1.

ab_norm

[out] 1-норма сбалансированной входной матрицы (максимум из сумм абсолютных значений элементов любой из колонок матрицы).

bb_norm

[out] 1-норма сбалансированной второй матрицы B.

rconde

[out] Вектор обратных чисел обусловленности для каждого собственного значения; рассчитывается если значение параметра sense равно 'E' или 'B'.

rcondv

[out] Вектор обратных чисел обусловленности для каждого правого собственного вектора; рассчитывается если значение параметра sense равно 'V' или 'B.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Балансировка входных матриц зависит от значения параметра *balance*.

ENUM_EIG_BALANCE

Перечисление, определяющее необходимость балансировки матриц.

Идентификатор	Описание
EIGBALANCE_N	Не проводится диагональное масштабирование и перестановка.
EIGBALANCE_P	Выполняются перестановки, чтобы привести матрицу к почти верхнетреугольному виду. Не выполняется диагональное масштабирование.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
EIGBALANCE_S	Выполняется диагональное масштабирование матрицы. Перестановка не проводится.
EIGBALANCE_B	Выполняется и диагональное масштабирование, и перестановка.

ENUM_EIG_VECTORS

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVECTORS_N	Вычисляются только собственные значения, без векторов.
EIGVECTORS_L	Вычисляются только левые собственные векторы.
EIGVECTORS_R	Вычисляются только правые собственные векторы.
EIGVECTORS_LR	Вычисляются левые и правые собственные векторы, собственные значения вычисляются всегда.

ENUM_EIG_SENSE

Перечисление, определяющее необходимость вычисления обратных чисел обусловленности.

Идентификатор	Описание
EIGSENSE_N	Не вычисляется ни одно из обратных чисел обусловленности.
EIGSENSE_E	Вычисляются только для собственных значений.
EIGSENSE_V	Вычисляются только для правых собственных векторов.
EIGSENSE_B	Вычисляется и для собственных значений, и для правых собственных векторов.

© 2000-2025, MetaQuotes Ltd.

EigenSolver2Shur

Вычисление для пары обычных квадратных матриц обобщённых собственных значений, обобщённых собственных векторов, обобщённых форм Шура и левых и правых векторов Шура (lapack-функция GGES).

Вычисляет обобщенные собственные значения, обобщенную действительную/комплексную форму Шура (S,T), опционально левую и/или правую матрицы векторов Шура (vsl и vsr) для пары действительных/комплексных несимметричных матриц (A,B) размером n на n. Это дает обобщенную факторизацию Шура:

(A,B) = (vsl*S *vsrH, vsl*T*vsrH)

Также может упорядочивать собственные значения таким образом, чтобы выбранный кластер собственных значений появлялся в ведущих диагональных блоках верхней квазитреугольной матрицы S и верхней треугольной матрицы T. Затем ведущие столбцы vsl и vsr образуют ортонормированный/унитарный базис для соответствующих левого и правого собственных пространств (дефляционных подпространств).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSolver2Shur(
   matrix&                B,         // вторая матрица в паре
   ENUM_EIG_VECTORS       jobvs,     // способ вычисления левых и правых векторов
   vector&                alpha,     // вектор рассчитанных собственных значений
   vector&                beta,      // вектор делителей собственных значений
   matrix&                shur_s,    // матрица S в форме Шура
   matrix&                shur_t,    // матрица T в форме Шура
   matrix&                vsl,       // матрица левых векторов Шура vsl
   matrix&                vsr        // матрица правых векторов Шура vsr
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSolver2Shur(
   matrixf&               B,         // вторая матрица в паре
   ENUM_EIG_VECTORS       jobvs,     // способ вычисления левых и правых векторов
   vectorcf&              alpha,     // вектор рассчитанных собственных значений
   vectorf&               beta,      // вектор делителей собственных значений
   matrixf&               shur_s,    // матрица S в форме Шура
   matrixf&               shur_t,    // матрица T в форме Шура
   matrixf&               vsl,       // матрица левых векторов Шура vsl
   matrixf&               vsr        // матрица правых векторов Шура vsr
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::EigenSolver2Shur(
   matrixc&               B,         // вторая матрица в паре
   ENUM_EIG_VECTORS       jobvs,     // способ вычисления левых и правых векторов
   vectorc&               alpha,     // вектор рассчитанных собственных значений
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorc&                beta,      // вектор делителей собственных значений
   matrixc&                shur_s,    // матрица S в форме Шура
   matrixc&                shur_t,    // матрица T в форме Шура
   matrixc&                vsl,       // матрица левых векторов Шура vsl
   matrixc&                vsr        // матрица правых векторов Шура vsr
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSolver2Shur(
   matrixcf&               B,         // вторая матрица в паре
   ENUM_EIG_VECTORS        jobvs,     // способ вычисления левых и правых векторов
   vectorcf&               alpha,     // вектор рассчитанных собственных значений
   vectorcf&               beta,      // вектор делителей собственных значений
   matrixcf&               shur_s,    // матрица S в форме Шура
   matrixcf&               shur_t,    // матрица T в форме Шура
   matrixcf&               vsl,       // матрица левых векторов Шура vsl
   matrixcf&               vsr        // матрица правых векторов Шура vsr
   );
```

Параметры

B

[out] Вторая матрица в паре.

jobvs

[in] Значение из перечисления ENUM_EIG_VECTORS, определяющее способ вычисления левых и правых собственных векторов.

alpha

[out] Вектор собственных значений.

beta

[out] Вектор делителей собственных значений.

shur_s

[out] Матрица S, блочная верхнетреугольная матрица Шура (форма Шура для входной матрицы).

shur_t

[out] Матрица T, блочная верхнетреугольная матрица Шура (форма Шура для второй матрицы в паре).

vsl

[out] Матрица левых векторов Шура.

vsr

[out] Матрица правых векторов Шура.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значений параметра *jobvs*.

Вторая матрица в паре должна быть того же самого размера, что и первая (входная) матрица.

Реальные (не комплексные) матрицы могут иметь комплексное решение. Поэтому вектор собственных значений должен быть комплексным. В случае комплексного решения код ошибки устанавливается в 4019 (ERR_MATH_OVERFLOW). В противном случае нужно учитывать только реальные части комплексных значений вектора собственных значений.

EigenSolverShur

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVECTORS_N	Вычисляются только собственные значения, без векторов.
EIGVECTORS_L	Вычисляются только левые собственные векторы.
EIGVECTORS_R	Вычисляются только правые собственные векторы.
EIGVECTORS_LR	Вычисляются левые и правые собственные векторы, собственные значения вычисляются всегда.

© 2000-2025, MetaQuotes Ltd.

EigenSolver2Blocked

Вычисление обобщённых собственных значений и собственных векторов для пары обычных квадратных матриц при помощи блочного алгоритма (lapack-функция GGEV3). Обе матрицы должны быть одинакового размера. Параметры метода точно такие же, как и у EigenSolver2.

Вычисления для типа matrix<double>

```
bool  matrix::EigenSolver2Blocked(
   matrix&               B,                    // вторая матрица в паре
   ENUM_EIG_VECTORS      jobv,                 // способ вычисления левых и правых
   vectorc&              alpha,                // вектор рассчитанных собственных з
   vector&               beta,                 // вектор делителей собственных знач
   matrix&               left_eigenvectors,    // матрица рассчитанных левых векто
   matrix&               right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSolver2Blocked(
   matrix&               B,                    // вторая матрица в паре
   ENUM_EIG_VECTORS      jobv,                 // способ вычисления левых и правых
   vectorcf&             alpha,                // вектор рассчитанных собственных з
   vectorf&              beta,                 // вектор делителей собственных знач
   matrixf&              left_eigenvectors,    // матрица рассчитанных левых векто
   matrixf&              right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::EigenSolver2Blocked(
   matrixc&              B,                    // вторая матрица в паре
   ENUM_EIG_VECTORS      jobv,                 // способ вычисления левых и правых
   vectorc&              alpha,                // вектор рассчитанных собственных з
   vectorc&              beta,                 // вектор делителей собственных знач
   matrixc&              left_eigenvectors,    // матрица рассчитанных левых векто
   matrixc&              right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSolver2(
   matrixcf&             B,                    // вторая матрица в паре
   ENUM_EIG_VECTORS      jobv,                 // способ вычисления левых и правых
   vectorcf&             alpha,                // вектор рассчитанных собственных з
   vectorcf&             beta,                 // вектор делителей собственных знач
   matrixcf&             left_eigenvectors,    // матрица рассчитанных левых векто
   matrixcf&             right_eigenvectors    // матрица рассчитанных правых векто
   );
```

Параметры

© 2000-2025, MetaQuotes Ltd.

B

[out] Вторая матрица в паре.

jobv

[in] Значение из перечисления ENUM_EIG_VECTORS, определяющее способ вычисления левых и правых собственных векторов.

alpha

[out] Вектор собственных значений.

beta

[out] Вектор делителей собственных значений.

left_eigenvectors

[out] Матрица левых собственных векторов.

righeft_eigenvectors

[out] Матрица правых собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха или false в случае ошибки.

Примечание

Расчёт зависит от значений параметра jobv.

Обобщённое собственное значение для пары матриц (A,B) — это скалярное λ или такое отношение alpha/beta = lambda, при котором A - lambda*B является сингулярным. Обычно представляется в виде пары (alpha,beta), поскольку существует интерпретация для beta=0 и даже для alpha=0 и beta=0.

Правый собственный вектор v(j), соответствующий собственному значению lambda(j) матрицы (A,B), удовлетворяет условию:

A * v(j) = lambda(j) * B * v(j).

Левый собственный вектор u(j), соответствующий собственному значению lambda(j) матрицы (A,B), удовлетворяет условию:

u(j)**H * A = lambda(j) * u(j)**H * B .

где u(j)**H — сопряженно-транспонированное значение u(j).

Реальные (не комплексные) матрицы могут иметь комплексное решение. Поэтому входной вектор собственных значений должен быть комплексным. В случае комплексного решения код ошибки устанавливается в 4019 (ERR_MATH_OVERFLOW). В противном случае нужно учитывать только реальные части комплексных значений вектора собственных значений.

ENUM_EIG_VECTORS

Перечисление, определяющее необходимость вычисления собственных векторов.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
EIGVECTORS_N	Вычисляются только собственные значения, без векторов.
EIGVECTORS_L	Вычисляются только левые собственные векторы.
EIGVECTORS_R	Вычисляются только правые собственные векторы.
EIGVECTORS_LR	Вычисляются левые и правые собственные векторы, собственные значения вычисляются всегда.

© 2000-2025, MetaQuotes Ltd.

EigenSolver2ShurBlocked

Вычисление для пары обычных квадратных матриц обобщённых собственных значений, обобщённых собственных векторов, обобщённых форм Шура и левых и правых векторов Шура при помощи блочного алгоритма (lapack-функция GGES3).

Вычисляет обобщенные собственные значения, обобщенную действительную/комплексную форму Шура (S,T), опционально левую и/или правую матрицы векторов Шура (vsl и vsr) для пары действительных/комплексных несимметричных матриц (A,B) размером n на n. Это дает обобщенную факторизацию Шура:

(A,B) = (vsl*S *vsrH, vsl*T*vsrH)

Также может упорядочивать собственные значения таким образом, чтобы выбранный кластер собственных значений появлялся в ведущих диагональных блоках верхней квазитреугольной матрицы S и верхней треугольной матрицы T. Затем ведущие столбцы vsl и vsr образуют ортонормированный/унитарный базис для соответствующих левого и правого собственных пространств (дефляционных подпространств).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrix&                B,         // вторая матрица в паре
   ENUM_EIG_VECTORS       jobvs,     // способ вычисления левых и правых векторов
   vectorc&               alpha,     // вектор рассчитанных собственных значений
   vector&                beta,      // вектор делителей собственных значений
   matrix&                shur_s,    // матрица S в форме Шура
   matrix&                shur_t,    // матрица T в форме Шура
   matrix&                vsl,       // матрица левых векторов Шура VSL
   matrix&                vsr        // матрица правых векторов Шура VSR
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrixf&               B,         // вторая матрица в паре
   ENUM_EIG_VECTORS       jobvs,     // способ вычисления левых и правых векторов
   vectorcf&              alpha,     // вектор рассчитанных собственных значений
   vectorf&               beta,      // вектор делителей собственных значений
   matrixf&               shur_s,    // матрица S в форме Шура
   matrixf&               shur_t,    // матрица T в форме Шура
   matrixf&               vsl,       // матрица левых векторов Шура VSL
   matrixf&               vsr        // матрица правых векторов Шура VSR
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrixc&               B,         // вторая матрица в паре
   ENUM_EIG_VECTORS       jobvs,     // способ вычисления левых и правых векторов
   vectorc&               alpha,     // вектор рассчитанных собственных значений
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorc&                beta,      // вектор делителей собственных значений
   matrixc&                shur_s,    // матрица S в форме Шура
   matrixc&                shur_t,    // матрица T в форме Шура
   matrixc&                vsl,       // матрица левых векторов Шура VSL
   matrixc&                vsr        // матрица правых векторов Шура VSR
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::EigenSolver2ShurBlocked(
   matrixcf&               B,         // вторая матрица в паре
   ENUM_EIG_VECTORS        jobvs,     // способ вычисления левых и правых векторов
   vectorcf&               alpha,     // вектор рассчитанных собственных значений
   vectorcf&               beta,      // вектор делителей собственных значений
   matrixcf&               shur_s,    // матрица S в форме Шура
   matrixcf&               shur_t,    // матрица T в форме Шура
   matrixcf&               vsl,       // матрица левых векторов Шура VSL
   matrixcf&               vsr        // матрица правых векторов Шура VSR
   );
```

Параметры

B

[in] Вторая матрица в паре.

jobvs

[in] Значение из перечисления ENUM_EIG_VECTORS, определяющее способ вычисления левых и правых собственных векторов.

alpha

[out] Вектор собственных значений.

beta

[out] Вектор делителей собственных значений.

shur_s

[out] Матрица S, блочная верхнетреугольная матрица Шура (форма Шура для входной матрицы).

shur_t

[out] Матрица T, блочная верхнетреугольная матрица Шура (форма Шура для второй матрицы в паре).

vsl

[out] Матрица левых векторов Шура VSL.

vsr

[out] Матрица правых векторов Шура VSR.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успеха или false в случае ошибки.

Примечание

Расчёт зависит от значений параметра *jobvs*.

Вторая матрица в паре должна быть того же самого размера, что и первая (входная) матрица.

Реальные (не комплексные) матрицы могут иметь комплексное решение. Поэтому вектор собственных значений должен быть комплексным. В случае комплексного решения код ошибки устанавливается в 4019 (ERR_MATH_OVERFLOW). В противном случае нужно учитывать только реальные части комплексных значений вектора собственных значений.

EigenSolverShur

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVECTORS_N	Вычисляются только собственные значения, без векторов.
EIGVECTORS_L	Вычисляются только левые собственные векторы.
EIGVECTORS_R	Вычисляются только правые собственные векторы.
EIGVECTORS_LR	Вычисляются левые и правые собственные векторы, собственные значения вычисляются всегда.

© 2000-2025, MetaQuotes Ltd.

Symmetric Matrices

Функции для вычисления собственных значений и собственных векторов симметричных или эрмитовых матриц с использованием алгоритма "разделяй и властвуй" (divide and conquer), что делает процесс эффективным и быстрым. Эти методы могут применяться для работы с матрицами разных типов данных, включая действительные и комплексные числа.

Функция	Действие
EigenSymmetricDC	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма "divide and conquere" (lapack-функции SYEVD, HEEVD).
EigenSymmetricQR	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи классического алгоритма QR (lapack-функции SYEV, HEEV).
EigenSymmetricRobust	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма Multiple Relatively Robust Representations, MRRR (lapack-функции SYEVR, HEEVR).
EigenSymmetricBisect	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма bisection (lapack-функции SYEVX, HEEVX).
EigenSymmetricDC2s	Вычисление всех собственных значений и, при необходимости, собственных векторов вещественной симметричной или эрмитовой (комплексно-сопряжённой) матрицы с использованием двухэтапного алгоритма приведения к тридиагональному виду. Если требуется вычисление собственных векторов, применяется алгоритм divide and conquer («разделяй и властвуй») (функции lapack SYEVD_2STAGE, HEEVD_2STAGE).
EigenSymmetricQR2s	Вычисление всех собственных значений и, при необходимости, собственных векторов вещественной симметричной или эрмитовой (комплексно-сопряжённой) матрицы с использованием двухэтапного алгоритма приведения к тридиагональному виду (функции lapack SYEV_2STAGE, HEEV_2STAGE).

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
EigenSymmetricRobust2s	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи двухэтапного алгоритма: сначала приведение к тридиагональному виду, затем применение алгоритма множественных относительно надежных представлений (Multiple Relatively Robust Representations, MRRR) (функции lapack SYEVR_2STAGE, HEEVR_2STAGE).
EigenSymmetricBisect2s	Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи двухэтапного алгоритма: сначала приведение к тридиагональному виду, затем применение метода бисекции (функции lapack SYEVX_2STAGE, HEEVX_2STAGE).
EigenSymmetric2DC	Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.
EigenSymmetric2QR	Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.
EigenSymmetric2Bisect	Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricDC

Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма "divide and conquere" (lapack-функции SYEVD, HEEVD).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobz,      // необходимость вычисления собственных векторов
   double&               EV,        // вектор рассчитанных собственных значений
   matrix&               V          // матрица рассчитанных собственных векторов
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobz,      // необходимость вычисления собственных векторов
   float&                EV,        // вектор рассчитанных собственных значений
   matrixf&              V          // матрица рассчитанных собственных векторов
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobz,      // необходимость вычисления собственных векторов
   double&               EV,        // вектор рассчитанных собственных значений
   matrixc&              V          // матрица рассчитанных собственных векторов
   );
```

Параметры

jobz

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления левых и правых собственных векторов.

EV

[out] Вектор собственных значений.

VL

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха или false в случае ошибки.

Примечание

Расчёт зависит от значений параметра jobz.

При значении EIGVALUES_V рассчитываются собственные векторы и собственные значения.

При значении EIGVALUES_N собственные векторы не рассчитываются. Рассчитываются только собственные значения.

© 2000-2025, MetaQuotes Ltd.

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricQR

Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи классического алгоритма QR (LAPACK-функции SYEV, HEEV).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   vector&               eigen_values,      // вектор собственных значений
   matrix&               eigen_vectors      // матрица собственных векторов
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   vectorf&              eigen_values,      // вектор собственных значений
   matrixf&              eigen_vectors      // матрица собственных векторов
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   vector&               eigen_values,      // вектор собственных значений
   matrixc&              eigen_vectors      // матрица собственных векторов
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   vectorf&              eigen_values,      // вектор собственных значений
   matrixcf&             eigen_vectors      // матрица собственных векторов
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Примечание

Расчет зависит от значений параметра jobv.

При значении jobv = EIGVALUES_V рассчитываются собственные векторы и собственные значения.

При значении EIGVALUES_N собственные векторы не рассчитываются. Рассчитываются только собственные значения.

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricRobust

Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма Multiple Relatively Robust Representations, MRRR (LAPACK-функции SYEVR, HEEVR).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер абсолютной допустимой ошибки
   vector&               eigen_values,      // вектор собственных значений
   matrix&               eigen_vectors      // матрица собственных векторов
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   float                 lower,             // нижняя граница подмножества
   float                 upper,             // верхняя граница подмножества
   float                 abstol,            // размер абсолютной допустимой ошибки
   vectorf&              eigen_values,      // вектор собственных значений
   matrixf&              eigen_vectors      // матрица собственных векторов
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер абсолютной допустимой ошибки
   vector&               eigen_values,      // вектор собственных значений
   matrixc&              eigen_vectors      // матрица собственных векторов
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   float                 lower,             // нижняя граница подмножества
   float                 upper,             // верхняя граница подмножества
   float                 abstol,            // размер абсолютной допустимой ошибки
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf&              eigen_values,       // вектор собственных значений
   matrixcf&             eigen_vectors       // матрица собственных векторов
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых собственных значений и векторов.

lower

[in] Нижняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

upper

[in] Верхняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

abstol

[in] Размер абсолютной допустимой ошибки.

Размер абсолютной допустимой ошибки, с которой требуется определить каждое собственное значение/собственный вектор.

Если jobv = 'V', то собственные значения и собственные векторы на выходе имеют нормы невязок, ограниченные значением abstol, и при этом скалярные произведения между различными собственными векторами также ограничены значением abstol.

Если abstol < n *eps*|T|, тогда используется значение n *eps*|T|, где eps — это машинная точность, а |T| — это 1-норма матрицы T. Собственные значения вычисляются с точностью eps*|T| независимо от abstol.

В случае, если важна высокая относительная точность, необходимо установить abstol в безопасное минимальное значение X такое, что 1.0/X не переполняется.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значения параметров *jobv* и *range*.

При значении BLASRANGE_A все собственные значения вычисляются, при этом параметры lower и upper игнорируются.

© 2000-2025, MetaQuotes Ltd.

При значении BLASRANGE_V вычисляются только те собственные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper

При значении BLASRANGE_I вычисляются только те собственные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три собственные значения.

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

ENUM_BLAS_RANGE

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
BLASRANGE_A	Будут найдены все сингулярные или собственные значения.
BLASRANGE_V	Будут найдены все сингулярные или собственные значения в полуоткрытом интервале (VL,VU].
BLASRANGE_I	Вычисляются особые или собственные значения с IL-го по IU-е.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricBisect

Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи алгоритма bisection (LAPACK-функции SYEVX, HEEVX).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер абсолютной допустимой ошибки
   vector&               eigen_values,      // вектор собственных значений
   matrix&               eigen_vectors      // матрица собственных векторов
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   float                 lower,             // нижняя граница подмножества
   float                 upper,             // верхняя граница подмножества
   float                 abstol,            // размер абсолютной допустимой ошибки
   vectorf&              eigen_values,      // вектор собственных значений
   matrixf&              eigen_vectors      // матрица собственных векторов
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер абсолютной допустимой ошибки
   vector&               eigen_values,      // вектор собственных значений
   matrixc&              eigen_vectors      // матрица собственных векторов
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // нужно ли вычислять собственные векторы
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   float                 lower,             // нижняя граница подмножества
   float                 upper,             // верхняя граница подмножества
   float                 abstol,            // размер абсолютной допустимой ошибки
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf&              eigen_values,       // вектор собственных значений
   matrixcf&             eigen_vectors       // матрица собственных векторов
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых собственных значений и векторов.

lower

[in] Нижняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

upper

[in] Верхняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

abstol

[in] Размер абсолютной допустимой ошибки.

Размер абсолютной допустимой ошибки, с которой требуется определить каждое собственное значение/собственный вектор.

Если jobv = 'V', то собственные значения и собственные векторы на выходе имеют нормы невязок, ограниченные значением abstol, и при этом скалярные произведения между различными собственными векторами также ограничены значением abstol.

Если abstol < n *eps*|T|, тогда используется значение n *eps*|T|, где eps — это машинная точность, а |T| — это 1-норма матрицы T. Собственные значения вычисляются с точностью eps*|T| независимо от abstol.

В случае, если важна высокая относительная точность, необходимо установить abstol в безопасное минимальное значение X такое, что 1.0/X не переполняется.

eigen_values

[out] Вектор собственных значений.

V

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значения параметров *jobv* и *range*.

При значении BLASRANGE_A все собственные значения вычисляются, при этом параметры lower и upper игнорируются.

© 2000-2025, MetaQuotes Ltd.

При значении BLASRANGE_V вычисляются только те собственные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper

При значении BLASRANGE_I вычисляются только те собственные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три собственные значения.

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

ENUM_BLAS_RANGE

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
BLASRANGE_A	Будут найдены все сингулярные или собственные значения.
BLASRANGE_V	Будут найдены все сингулярные или собственные значения в полуоткрытом интервале (VL,VU].
BLASRANGE_I	Вычисляются особые или собственные значения с IL-го по IU-е.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricDC2s

Вычисление всех собственных значений и, при необходимости, собственных векторов вещественной симметричной или эрмитовой (комплексно-сопряжённой) матрицы с использованием двухэтапного алгоритма приведения к тридиагональному виду. Если требуется вычисление собственных векторов, применяется алгоритм divide and conquer («разделяй и властвуй») (функции lapack SYEVD_2STAGE, HEEVD_2STAGE).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrix&               eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   vectorf&              eigen_values,      // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrixc&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   vectorf&              eigen_values,      // вектор рассчитанных собственных значен
   matrixcf&             eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значений параметра jobv.

При значении jobv = EIGVALUES_V рассчитываются собственные векторы и собственные значения. В текущей реализации OpenBLAS использование этого значения недоступно. В случае его использования будет возвращена ошибка 4003 (ERR_INVALID_PARAMETER).

При значении EIGVALUES_N собственные векторы не рассчитываются. Рассчитываются только собственные значения.

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения. Недоступно в этой версии.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricQR2s

Вычисление всех собственных значений и, при необходимости, собственных векторов вещественной симметричной или эрмитовой (комплексно-сопряжённой) матрицы с использованием двухэтапного алгоритма приведения к тридиагональному виду (функции lapack SYEV_2STAGE, HEEV_2STAGE).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrix&               eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   vectorf&              eigen_values,      // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrixc&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   vectorf&              eigen_values,      // вектор рассчитанных собственных значен
   matrixcf&             eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значений параметра jobv.

При значении jobv = EIGVALUES_V рассчитываются собственные векторы и собственные значения. В текущей реализации OpenBLAS использование этого значения недоступно. В случае его использования будет возвращена ошибка 4003 (ERR_INVALID_PARAMETER).

При значении EIGVALUES_N собственные векторы не рассчитываются. Рассчитываются только собственные значения.

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения. Недоступно в этой версии.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricRobust2s

Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи двухэтапного алгоритма: сначала приведение к тридиагональному виду, затем применение алгоритма множественных относительно надежных представлений (Multiple Relatively Robust Representations, MRRR) (функции lapack SYEVR_2STAGE, HEEVR_2STAGE).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер допустимой абсолютной ошибки (а
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrix&               eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   float                 lower,             // нижняя граница подмножества
   float                 upper,             // верхняя граница подмножества
   float                 abstol,            // размер допустимой абсолютной ошибки (а
   vectorf&              eigen_values,      // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер допустимой абсолютной ошибки (а
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrixc&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   float                 lower,             // нижняя граница подмножества
```

© 2000-2025, MetaQuotes Ltd.

```
   float                upper,              // верхняя граница подмножества
   float                abstol,             // размер допустимой абсолютной ошибки (а
   vectorf&             eigen_values,       // вектор рассчитанных собственных значен
   matrixcf&            eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых собственных значений и векторов.

lower

[in] Нижняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

upper

[in] Верхняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

abstol

[in] Размер абсолютной допустимой ошибки.

Размер абсолютной допустимой ошибки, с которой требуется определить каждое собственное значение/собственный вектор.

Если jobv = 'V', то собственные значения и собственные векторы на выходе имеют нормы невязок, ограниченные значением abstol, и при этом скалярные произведения между различными собственными векторами также ограничены значением abstol.

Если abstol < n *eps*|T|, тогда используется значение n *eps*|T|, где eps — это машинная точность, а |T| — это 1-норма матрицы T. Собственные значения вычисляются с точностью eps*|T| независимо от abstol.

В случае, если важна высокая относительная точность, необходимо установить abstol в безопасное минимальное значение X такое, что 1.0/X не переполняется.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха или false в случае ошибки.

Примечание

Расчёт зависит от значения параметров *jobv* и *range*.

© 2000-2025, MetaQuotes Ltd.

При значении jobv = EIGVALUES_V рассчитываются собственные векторы и собственные значения. В текущей реализации OpenBLAS использование этого значения недоступно. В случае его использования будет возвращена ошибка 4003 (ERR_INVALID_PARAMETER).

При значении EIGVALUES_N собственные векторы не рассчитываются. Рассчитываются только собственные значения.

При значении BLASRANGE_A все собственные значения вычисляются, при этом параметры lower и upper игнорируются.

При значении BLASRANGE_V вычисляются только те собственные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper

При значении BLASRANGE_I вычисляются только те собственные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три собственные значения.

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения. Недоступно в этой версии.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

ENUM_BLAS_RANGE

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
BLASRANGE_A	Будут найдены все сингулярные или собственные значения.
BLASRANGE_V	Будут найдены все сингулярные или собственные значения в полуоткрытом интервале (VL,VU].
BLASRANGE_I	Вычисляются особые или собственные значения с IL-го по IU-е.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetricBisect2s

Вычисление собственных значений и собственных векторов симметричной или эрмитовой (комплексно-сопряжённой) матрицы при помощи двухэтапного алгоритма: сначала приведение к тридиагональному виду, затем применение метода бисекции (функции lapack SYEVX_2STAGE, HEEVX_2STAGE).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер допустимой абсолютной ошибки (
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrix&               eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных
   float                 lower,             // нижняя граница подмножества
   float                 upper,             // верхняя граница подмножества
   float                 abstol,            // размер допустимой абсолютной ошибки (
   vectorf&              eigen_values,      // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер допустимой абсолютной ошибки (
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrixc&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных
   float                 lower,             // нижняя граница подмножества
   float                 upper,             // верхняя граница подмножества
```

© 2000-2025, MetaQuotes Ltd.

```
   float                abstol,             // размер допустимой абсолютной ошибки (a
   vectorf&             eigen_values,       // вектор рассчитанных собственных значен
   matrixcf&            eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых собственных значений и векторов.

lower

[in] Нижняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

upper

[in] Верхняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

abstol

[in] Размер абсолютной допустимой ошибки.

Размер абсолютной допустимой ошибки, с которой требуется определить каждое собственное значение/собственный вектор.

Если jobv = 'V', то собственные значения и собственные векторы на выходе имеют нормы невязок, ограниченные значением abstol, и при этом скалярные произведения между различными собственными векторами также ограничены значением abstol.

Если abstol < n *eps*|T|, тогда используется значение n *eps*|T|, где eps — это машинная точность, а |T| — это 1-норма матрицы T. Собственные значения вычисляются с точностью eps*|T| независимо от abstol.

В случае, если важна высокая относительная точность, необходимо установить abstol в безопасное минимальное значение X такое, что 1.0/X не переполняется.

eigen_values

[out] Вектор собственных значений.

V

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха или false в случае ошибки.

Примечание

Расчёт зависит от значения параметров *jobv* и *range*.

© 2000-2025, MetaQuotes Ltd.

При значении jobv = EIGVALUES_V рассчитываются собственные векторы и собственные значения. В текущей реализации OpenBLAS использование этого значения недоступно. В случае его использования будет возвращена ошибка 4003 (ERR_INVALID_PARAMETER).

При значении EIGVALUES_N собственные векторы не рассчитываются. Рассчитываются только собственные значения.

При значении BLASRANGE_A все собственные значения вычисляются, при этом параметры lower и upper игнорируются.

При значении BLASRANGE_V вычисляются только те собственные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper

При значении BLASRANGE_I вычисляются только те собственные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три собственные значения.

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения. Недоступно в этой версии.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

ENUM_BLAS_RANGE

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
BLASRANGE_A	Будут найдены все сингулярные или собственные значения.
BLASRANGE_V	Будут найдены все сингулярные или собственные значения в полуоткрытом интервале (VL,VU].
BLASRANGE_I	Вычисляются особые или собственные значения с IL-го по IU-е.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetric2DC

Вычисление всех собственных значений и, при необходимости, собственных векторов для обобщенной симметричной задачи на собственные значения: A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

Здесь предполагается, что матрицы A и B — симметричные (или эрмитовы), а также что матрица B положительно определенаа. Если требуется вычисление собственные векторы, применяется алгоритм divide and conquer («разделяй и властвуй») (функции LAPACK SYGVD и HEGVD).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // тип задачи
   ENUM_EIG_VALUES       jobv,               // нужно ли вычислять собственные векторы
   matrix&               B,                  // вторая матрица
   vector&               eigen_values,       // вектор собственных значений
   matrix&               eigen_vectors,      // матрица собственных векторов
   matrix&               triangular_factor   // треугольный множитель из разложения Хо
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // тип задачи
   ENUM_EIG_VALUES       jobv,               // нужно ли вычислять собственные векторы
   matrixf&              B,                  // вторая матрица
   vectorf&              eigen_values,       // вектор собственных значений
   matrixf&              eigen_vectors,      // матрица собственных векторов
   matrixf&              triangular_factor   // треугольный множитель из разложения Хо
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // тип задачи
   ENUM_EIG_VALUES       jobv,               // нужно ли вычислять собственные векторы
   matrixc&              B,                  // вторая матрица
   vector&               eigen_values,       // вектор собственных значений
   matrixc&              eigen_vectors,      // матрица собственных векторов
   matrixc&              triangular_factor   // треугольный множитель из разложения Хо
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // тип задачи
   ENUM_EIG_VALUES       jobv,               // нужно ли вычислять собственные векторы
   matrixcf&             B,                  // вторая матрица
   vectorf&              eigen_values,       // вектор собственных значений
   matrixcf&             eigen_vectors,      // матрица собственных векторов
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixcf&             triangular_factor   // треугольный множитель из разложения Хо
   );
```

Параметры

itype

[in] Значение перечисления ENUM_EIGS2_TYPE, определяющее тип задачи: A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

B

[in] Вторая матрица B. Должна быть положительно определенной симметричной (или эрмитово сопряженной) матрицей.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

triangular_factor

[out] Треугольный множитель (верхний или нижний) из разложения Холецкого B = U**T*U или B = L*L**T.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными). Вторая матрица B должна быть положительно определенной симметричной. Если входная матрица и вторая матрица B треугольные, то обе они должны быть одинаковой треугольной формы — верхней или нижней.

ENUM_EIGS2_TYPE

Перечисление, которое определяет тип задачи.

Идентификатор	Описание
EIGS2TYPE_1	1: A*x = (lambda)*B*x
EIGS2TYPE_2	2: A*B*x = (lambda)*x
EIGS2TYPE_3	3: B*A*x = (lambda)*x

ENUM_EIG_VALUES

© 2000-2025, MetaQuotes Ltd.

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetric2QR

Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.

Предполагается, что матрицы A и B симметричны (или эрмитовы), а матрица B также положительно определена. Метод использует QR-алгоритм (функции LAPACK SYGV, HEGV).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // тип решаемой задачи
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных в
   matrix&               B,                  // вторая матрица
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrix&               eigen_vectors,      // матрица рассчитанных собственных векто
   matrix&               triangular_factor   // треугольный множитель из разложения Хо
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // тип решаемой задачи
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных в
   matrixf&              B,                  // вторая матрица
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors,      // матрица рассчитанных собственных векто
   matrixf&              triangular_factor   // треугольный множитель из разложения Хо
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // тип решаемой задачи
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных в
   matrixc&              B,                  // вторая матрица
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrixc&              eigen_vectors,      // матрица рассчитанных собственных векто
   matrixc&              triangular_factor   // треугольный множитель из разложения Хо
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // тип решаемой задачи
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных в
   matrixcf&             B,                  // вторая матрица
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixcf&             eigen_vectors,      // матрица рассчитанных собственных векто
   matrixcf&             triangular_factor   // треугольный множитель из разложения Хо
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Параметры

itype

[in] Значение перечисления ENUM_EIGS2_TYPE, указывающее тип решаемой задачи: A*x=(lambda)*B*x, A*Bx=(lambda)*x или or B*A*x=(lambda)*x.

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

B

[in] Вторая матрица B. Должна быть положительно определенной симметричной (или эрмитово сопряженной) матрицей.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

triangular_factor

[out] Треугольный множитель U или L из разложения Холецкого = U**T*U или B = L*L**T.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово сопряженными). Вторая матрица B должна быть положительно определенной симметричной матрицей. Если входная матрица и вторая матрица B являются треугольными, то обе должны быть одного типа: верхнетреугольными или нижнетрегольными.

ENUM_EIGS2_TYPE

Перечисление, определяющее тип решаемой задачи.

Идентификатор	Описание
EIGS2TYPE_1	1: A*x = (lambda)*B*x
EIGS2TYPE_2	2: A*B*x = (lambda)*x
EIGS2TYPE_3	3: B*A*x = (lambda)*x

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

© 2000-2025, MetaQuotes Ltd.

EigenSymmetric2Bisect

Вычисление всех собственных значений и, при необходимости, собственных векторов обобщенной симметрично определенной собственной задачи вида A*x=(lambda)*B*x, A*Bx=(lambda)*x или B*A*x=(lambda)*x.

Предполагается, что матрицы A и B симметричны (или эрмитовы), а матрица B также положительно определена. Собственные значения и собственные векторы можно выбрать, указав либо диапазон значений, либо диапазон индексов для желаемых собственных значений. Метод использует алгоритм бисекции (функции LAPACK: SYGVX, HEGVX).

Вычисления для типа matrix<double>

```
bool  matrix::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // тип решаемой задачи
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,              // подмножество вычисляемых собственных з
   double                lower,              // нижняя граница подмножества
   double                upper,              // верхняя граница подмножества
   double                abstol,             // размер допустимой абсолютной ошибки (а
   matrix&               B,                  // вторая матрица
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrix&               eigen_vectors,      // матрица рассчитанных собственных векто
   matrix&               triangular_factor   // треугольный множитель из разложения Хо
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // тип решаемой задачи
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,              // подмножество вычисляемых собственных з
   float                 lower,              // нижняя граница подмножества
   float                 upper,              // верхняя граница подмножества
   float                 abstol,             // размер допустимой абсолютной ошибки (а
   matrixf&              B,                  // вторая матрица
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors,      // матрица рассчитанных собственных векто
   matrixf&              triangular_factor   // треугольный множитель из разложения Хо
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // тип решаемой задачи
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,              // подмножество вычисляемых собственных з
   double                lower,              // нижняя граница подмножества
   double                upper,              // верхняя граница подмножества
   double                abstol,             // размер допустимой абсолютной ошибки (а
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixc&              B,                  // вторая матрица
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrixc&              eigen_vectors,      // матрица рассчитанных собственных векто
   matrixc&              triangular_factor   // треугольный множитель из разложения Хо
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // тип решаемой задачи
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,              // подмножество вычисляемых собственных з
   float                 lower,              // нижняя граница подмножества
   float                 upper,              // верхняя граница подмножества
   float                 abstol,             // размер допустимой абсолютной ошибки (a
   matrixcf&             B,                  // вторая матрица
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixcf&             eigen_vectors,      // матрица рассчитанных собственных векто
   matrixcf&             triangular_factor   // треугольный множитель из разложения Хо
   );
```

Параметры

itype

[in] Значение перечисления ENUM_EIGS2_TYPE, указывающее тип решаемой задачи: A*x=(lambda)*B*x, A*Bx=(lambda)*x или or B*A*x=(lambda)*x.

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых собственных значений и векторов.

lower

[in] Нижняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

upper

[in] Верхняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

abstol

[in] Размер абсолютной допустимой ошибки.

Размер абсолютной допустимой ошибки, с которой требуется определить каждое собственное значение/собственный вектор.

Если jobv = 'V', то собственные значения и собственные векторы на выходе имеют нормы невязок, ограниченные значением abstol, и при этом скалярные произведения между различными собственными векторами также ограничены значением abstol.

© 2000-2025, MetaQuotes Ltd.

Если abstol < n *eps*|T|, тогда используется значение n *eps*|T|, где eps — это машинная точность, а |T| — это 1-норма матрицы T. Собственные значения вычисляются с точностью eps*|T| независимо от abstol.

В случае, если важна высокая относительная точность, необходимо установить abstol в безопасное минимальное значение X такое, что 1.0/X не переполняется.

B

[in] Вторая матрица B. Должна быть положительно определенной симметричной (или эрмитово сопряженной) матрицей.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

triangular_factor

[out] Треугольный множитель U или L из разложения Холецкого = U**T*U или B = L*L**T.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчет зависит от значения параметров *jobv* и *range*.

При значении BLASRANGE_A вычисляются все собственные значения, при этом параметры lower и upper игнорируются.

При значении BLASRANGE_V вычисляются только те собственные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper

При значении BLASRANGE_I вычисляются только те собственные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три собственные значения.

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово сопряженными). Вторая матрица B должна быть положительно определенной симметричной матрицей. Если входная матрица и вторая матрица B являются треугольными, то обе должны быть одного типа: верхнетреугольными или нижнетрегольными.

ENUM_EIGS2_TYPE

Перечисление, определяющее тип решаемой задачи.

Идентификатор	Описание
EIGS2TYPE_1	1: A*x = (lambda)*B*x

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
EIGS2TYPE_2	2: A*B*x = (lambda)*x
EIGS2TYPE_3	3: B*A*x = (lambda)*x

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

ENUM_BLAS_RANGE

Перечисление, определяющее способ вычисления правых сингулярных векторов.

Идентификатор	Описание
BLASRANGE_A	Будут найдены все сингулярные или собственные значения.
BLASRANGE_V	Будут найдены все сингулярные или собственные значения в полуоткрытом интервале (VL,VU].
BLASRANGE_I	Вычисляются особые или собственные значения с IL-го по IU-е.

© 2000-2025, MetaQuotes Ltd.

Tridiagonal Matrices

Функции для вычисления собственных значений и собственных векторов симметричных трехдиагональных матриц с использованием различных алгоритмов. Каждая функция реализует определенный метод решения и поддерживает матрицы типа double и float.

Общие параметры:

- jobv — определяет необходимость вычислять собственные векторы (EIGVALUES_V) или только собственные значения (EIGVALUES_N).
- range — определяет диапазон вычисляемых значений (BLASRANGE_A, BLASRANGE_V, BLASRANGE_I).
- lower и upper — нижние и верхние границы для вычисления подмножеств спектра.
- abstol — размер допустимой абсолютной ошибки.

Все функции работают с симметричными трехдиагональными матрицами и позволяют выбрать наиболее подходящий алгоритм в зависимости от требований к производительности и точности.

Функция	Выполняемое действие
EigenTridiagonalDC	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием алгоритма "divide and conquer" (lapack-функция STEVD).
EigenTridiagonalQR	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием QR-алгоритма (lapack-функция STEV).
EigenTridiagonalRobust	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы при помощи алгоритма Multiple Relatively Robust Representations, MRRR (lapack-функция STEVR).
EigenTridiagonalBisect	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием алгоритма бисекции (lapack-функция STEVX).
EigenTridiagonalQL	Вычисление всех собственных значений симметричной трехдиагональной матрицы с использованием варианта QR-алгоритма без квадратного корня (lapack-функция STERF).
EigenTridiagonalDCQ	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием алгоритма "разделяй и властвуй" (lapack-функция STEDC).

© 2000-2025, MetaQuotes Ltd.

Функция	Выполняемое действие
EigenTridiagonalQRQ	Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием QR-алгоритма (lapack-функция STEQR).
EigenTridiagonalPosDefQ	Вычисление собственных значений и собственных векторов симметричной положительно определенной трехдиагональной матрицы с использованием QR-алгоритма (lapack-функция PTEQR).

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalDC

Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием алгоритма "divide and conquer" (lapack-функция STEVD).

Вычисления для типа matrix<double>

```
bool  matrix::EigenTridiagonalDC(
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrix&               eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenTridiagonalDC(
   ENUM_EIG_VALUES       jobv,               // необходимость вычисления собственных
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значений параметра jobv.

При значении jobv = EIGVALUES_V рассчитываются собственные векторы и собственные значения.

При значении EIGVALUES_N собственные векторы не рассчитываются. Рассчитываются только собственные значения.

На входе должна быть симметричная матрица в трехдиагональной форме.

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalQR

Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием QR-алгоритма (lapack-функция STEV).

Вычисления для типа matrix<double>

```
bool  matrix::EigenTridiagonalQR(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrix&               eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenTridiagonalQR(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных
   vectorf&              eigen_values,      // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значений параметра jobv.

При значении jobv = EIGVALUES_V рассчитываются собственные векторы и собственные значения.

При значении EIGVALUES_N собственные векторы не рассчитываются. Рассчитываются только собственные значения.

На входе должна быть симметричная матрица в трехдиагональной форме.

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalRobust

Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы при помощи алгоритма Multiple Relatively Robust Representations, MRRR (lapack-функция STEVR).

Вычисления для типа matrix<double>

```
bool  matrix::EigenTridiagonalRobust(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер допустимой абсолютной ошибки (а
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrix&               eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenTridiagonalRobust(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   float                 lower,             // нижняя граница подмножества
   float                 upper,             // верхняя граница подмножества
   float                 abstol,            // размер допустимой абсолютной ошибки (а
   vectorf&              eigen_values,      // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых собственных значений и векторов.

lower

[in] Нижняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

upper

[in] Верхняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

abstol

[in] Размер абсолютной допустимой ошибки.

© 2000-2025, MetaQuotes Ltd.

Размер абсолютной допустимой ошибки, с которой требуется определить каждое собственное значение/собственный вектор.

Если jobv = 'V', то собственные значения и собственные векторы на выходе имеют нормы невязок, ограниченные значением abstol, и при этом скалярные произведения между различными собственными векторами также ограничены значением abstol.

Если abstol < n *eps*|T|, тогда используется значение n *eps*|T|, где eps — это машинная точность, а |T| — это 1-норма матрицы T. Собственные значения вычисляются с точностью eps*|T| независимо от abstol.

В случае, если важна высокая относительная точность, необходимо установить abstol в безопасное минимальное значение X такое, что 1.0/X не переполняется.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значения параметров *jobv* и *range*.

При значении BLASRANGE_A все собственные значения вычисляются, при этом параметры lower и upper игнорируются.

При значении BLASRANGE_V вычисляются только те собственные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper

При значении BLASRANGE_I вычисляются только те собственные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три собственные значения.

На входе должна быть симметричная матрица в трехдиагональной форме.

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

ENUM_BLAS_RANGE

Перечисление, определяющее способ вычисления правых сингулярных векторов.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
BLASRANGE_A	Будут найдены все сингулярные или собственные значения.
BLASRANGE_V	Будут найдены все сингулярные или собственные значения в полуоткрытом интервале (VL,VU].
BLASRANGE_I	Вычисляются особые или собственные значения с IL-го по IU-е.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalBisect

Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием алгоритма бисекции (lapack-функция STEVX).

Вычисления для типа matrix<double>

```
bool  matrix::EigenTridiagonalBisect(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   double                lower,             // нижняя граница подмножества
   double                upper,             // верхняя граница подмножества
   double                abstol,            // размер допустимой абсолютной ошибки (а
   vector&               eigen_values,      // вектор рассчитанных собственных значен
   matrix&               eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenTridiagonalBisect(
   ENUM_EIG_VALUES       jobv,              // необходимость вычисления собственных в
   ENUM_BLAS_RANGE       range,             // подмножество вычисляемых собственных з
   float                 lower,             // нижняя граница подмножества
   float                 upper,             // верхняя граница подмножества
   float                 abstol,            // размер допустимой абсолютной ошибки (а
   vectorf&              eigen_values,      // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors      // матрица рассчитанных собственных векто
   );
```

Параметры

jobv

[in] Значение из перечисления ENUM_EIG_VALUES, определяющее способ вычисления собственных векторов.

range

[in] Значение из перечисления ENUM_BLAS_RANGE, определяющее подмножество вычисляемых собственных значений и векторов.

lower

[in] Нижняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

upper

[in] Верхняя граница подмножества собственных значений, задается в зависимости от значения параметра *range*.

abstol

[in] Размер абсолютной допустимой ошибки.

Размер абсолютной допустимой ошибки, с которой требуется определить каждое собственное значение/собственный вектор.

© 2000-2025, MetaQuotes Ltd.

Если jobv = 'V', то собственные значения и собственные векторы на выходе имеют нормы невязок, ограниченные значением abstol, и при этом скалярные произведения между различными собственными векторами также ограничены значением abstol.

Если abstol < n *eps*|T|, тогда используется значение n *eps*|T|, где eps — это машинная точность, а |T| — это 1-норма матрицы T. Собственные значения вычисляются с точностью eps*|T| независимо от abstol.

В случае, если важна высокая относительная точность, необходимо установить abstol в безопасное минимальное значение X такое, что 1.0/X не переполняется.

eigen_values

[out] Вектор собственных значений.

V

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчёт зависит от значения параметров *jobv* и *range*.

При значении BLASRANGE_A все собственные значения вычисляются, при этом параметры lower и upper игнорируются.

При значении BLASRANGE_V вычисляются только те собственные значения (и их векторы), которые попадают в диапазон вещественных значений, заданных параметрами lower и upper

При значении BLASRANGE_I вычисляются только те собственные значения (и их векторы), которые попадают в диапазон целочисленных индексов, заданных параметрами lower и upper. Например, при параметрах lower=0 и upper=2 вычисляются только первые три собственные значения.

На входе должна быть симметричная матрица в трехдиагональной форме.

ENUM_EIG_VALUES

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGVALUES_V	Вычисляются собственные векторы и собственные значения.
EIGVALUES_N	Вычисляются только собственные значения, без векторов.

ENUM_BLAS_RANGE

Перечисление, определяющее способ вычисления правых сингулярных векторов.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
BLASRANGE_A	Будут найдены все сингулярные или собственные значения.
BLASRANGE_V	Будут найдены все сингулярные или собственные значения в полуоткрытом интервале (VL,VU].
BLASRANGE_I	Вычисляются особые или собственные значения с IL-го по IU-е.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalQL

Вычисление всех собственных значений симметричной трехдиагональной матрицы с использованием варианта QR-алгоритма без квадратного корня (lapack-функция STERF).

Вычисления для типа matrix<double>

```
bool  matrix::EigenTridiagonalQL(
   vector&               eigen_values        // вектор вычисленных собственных значени
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenTridiagonalQL(
   vectorf&              eigen_values        // вектор вычисленных собственных значени
   );
```

Параметры

eigen_values

[out] Вектор собственных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

На вход должна подаваться симметричная матрица в трехдиагональной форме.

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalDCQ

Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием алгоритма "разделяй и властвуй" (lapack-функция STEDC). В отличие от метода EigenTridiagonalDC данный метод может быть использован для вычисления собственных векторов оригинальной симметричной матрицы. Симметричная матрица может быть преобразована (reduced) в трёхдиагональную форму методом ReduceSymmetricToTridiagonal. Ортогональная матрица Q, полученная в результате такого преобразования, используется для вычисления собственных векторов оригинальной симметричной матрицы.

Вычисления для типа matrix<double>

```
bool  matrix::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrix&               Q,                  // ортогональная матрица, используемая пр
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrix&               eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrixf&              Q,                  // ортогональная матрица, используемая пр
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrixc&              Q,                  // ортогональная матрица, используемая пр
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrixc&              eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrixcf&             Q,                  // ортогональная матрица, используемая пр
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixcf&             eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Параметры

compv

[in] Значение из перечисления ENUM_EIGTRIDIAG_Z, определяющее способ вычисления собственных векторов.

© 2000-2025, MetaQuotes Ltd.

Q

[in] Ортогональная матрица Q, полученная методом ReflectTridiagonalToQ.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчет зависит от значения параметра compv.

При compv = EIGCOMPZ_N рассчитываются только собственные значения, собственные векторы при этом не вычисляются.

При значении EIGCOMPZ_V вычисляются собственные значения, а также собственные векторы исходной симметричной матрицы.

Если параметр compv установлен в значение EIGCOMPZ_I, в дополнение к собственным значениям вычисляются собственные векторы трехдиагональной матрицы.

На вход должна подаваться симметричная матрица в трехдиагональной форме.

ENUM_EIGTRIDIAG_Z

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGCOMPZ_N	'N' — вычислять только собственные значения
EIGCOMPZ_V	'V' — дополнительно вычислять собственные векторы симметричной матрицы
EIGCOMPZ_I	'I' — дополнительно вычислять собственные векторы трехдиагональной матрицы

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalQRQ

Вычисление собственных значений и собственных векторов симметричной трехдиагональной матрицы с использованием QR-алгоритма (lapack-функция STEQR). В отличие от метода EigenTridiagonalQR данный метод может быть использован для вычисления собственных векторов оригинальной симметричной матрицы. Симметричная матрица может быть преобразована (reduced) в трёхдиагональную форму методом ReduceSymmetricToTridiagonal. Ортогональная матрица Q, полученная в результате такого преобразования, используется для вычисления собственных векторов оригинальной симметричной матрицы.

Вычисления для типа matrix<double>

```
bool  matrix::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrix&               Q,                  // ортогональная матрица, используемая пр
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrix&               eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrixf&              Q,                  // ортогональная матрица, используемая пр
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrixc&              Q,                  // ортогональная матрица, используемая пр
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrixc&              eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrixcf&             Q,                  // ортогональная матрица, используемая пр
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixcf&             eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Параметры

compv

[in] Значение из перечисления ENUM_EIGTRIDIAG_Z, определяющее способ вычисления собственных векторов.

© 2000-2025, MetaQuotes Ltd.

Q

[in] Ортогональная матрица Q, полученная методом ReflectTridiagonalToQ.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчет зависит от значения параметра compv.

При compv = EIGCOMPZ_N рассчитываются только собственные значения, собственные векторы при этом не вычисляются.

При значении EIGCOMPZ_V вычисляются собственные значения, а также собственные векторы исходной симметричной матрицы.

Если параметр compv установлен в значение EIGCOMPZ_I, в дополнение к собственным значениям вычисляются собственные векторы трехдиагональной матрицы.

На вход должна подаваться симметричная матрица в трехдиагональной форме.

ENUM_EIGTRIDIAG_Z

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGCOMPZ_N	'N' — вычислять только собственные значения
EIGCOMPZ_V	'V' — дополнительно вычислять собственные векторы симметричной матрицы
EIGCOMPZ_I	'I' — дополнительно вычислять собственные векторы трехдиагональной матрицы

© 2000-2025, MetaQuotes Ltd.

EigenTridiagonalPosDefQ

Вычисление собственных значений и собственных векторов симметричной положительно определенной трехдиагональной матрицы с использованием QR-алгоритма (lapack-функция PTEQR). При помощи данного метода могут быть вычислены собственные векторы как входной трехдиагональной матрицы, так и оригинальной положительно определенной симметричной матрицы. Для этого используется ортогональная матрица Q, полученная в результате приведения к трехдиагональной форме.

Вычисления для типа matrix<double>

```
bool  matrix::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrix&               Q,                  // ортогональная матрица, используемая пр
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrix&               eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrixf&              Q,                  // ортогональная матрица, используемая пр
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixf&              eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrixc&              Q,                  // ортогональная матрица, используемая пр
   vector&               eigen_values,       // вектор рассчитанных собственных значен
   matrixc&              eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // необходимость вычисления собственных в
   matrixcf&             Q,                  // ортогональная матрица, используемая пр
   vectorf&              eigen_values,       // вектор рассчитанных собственных значен
   matrixcf&             eigen_vectors       // матрица рассчитанных собственных векто
   );
```

Параметры

compv

[in] Значение из перечисления ENUM_EIGTRIDIAG_Z, определяющее способ вычисления собственных векторов.

Q

© 2000-2025, MetaQuotes Ltd.

[in] Ортогональная матрица Q, полученная методом ReflectTridiagonalToQ.

eigen_values

[out] Вектор собственных значений.

eigen_vectors

[out] Матрица собственных векторов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Расчет зависит от значения параметра compv.

При compv = EIGCOMPZ_N рассчитываются только собственные значения, собственные векторы при этом не вычисляются.

При значении EIGCOMPZ_V вычисляются собственные значения, а также собственные векторы исходной симметричной матрицы.

Если параметр compv установлен в значение EIGCOMPZ_I, в дополнение к собственным значениям вычисляются собственные векторы трехдиагональной матрицы.

На вход должна подаваться симметричная положительно определенная трехдиагональная матрица.

ENUM_EIGTRIDIAG_Z

Перечисление, определяющее необходимость вычисления собственных векторов.

Идентификатор	Описание
EIGCOMPZ_N	'N' — вычислять только собственные значения
EIGCOMPZ_V	'V' — дополнительно вычислять собственные векторы симметричной матрицы
EIGCOMPZ_I	'I' — дополнительно вычислять собственные векторы трехдиагональной матрицы

© 2000-2025, MetaQuotes Ltd.

Orthogonal Factorizations

OpenBLAS предоставляет ряд процедур для разложения общей прямоугольной матрицы A размера m на n в произведение ортогональной (в комплексном случае — унитарной) и треугольной (или, возможно, трапециевидной) матриц.

Реальная матрица Q называется ортогональной, если $Q^T Q = I$; комплексная матрица Q называется унитарной, если $Q^H Q = I$. Ортогональные и унитарные матрицы обладают важным свойством — они сохраняют евклидову норму вектора:

$||x||_2 = ||Qx||_2$, если Q ортогональна или унитарна

Благодаря этому такие матрицы способствуют численной устойчивости, поскольку не усиливают ошибки округления.

Ортогональные разложения широко применяются при решении задач наименьших квадратов. Их также можно использовать для выполнения предварительных шагов при решении задач на собственные значения или сингулярные значения.

Функция	Выполняемое действие
FactorizationQR	Вычисляет QR-разложение общей матрицы размера m на n: A = Q * R. LAPACK-функция GEQRF.
FactorizationQRNonNeg	Вычисляет QR-разложение общей матрицы размера m на n: A = Q * R, где R — верхнетреугольная матрица с неотрицательными элементами на диагонали. LATPACK-функция GEQRFP.
FactorizationQRPivot	Вычисляет QR-разложение общей матрицы размера m на n с перестановкой столбцов: A * P = Q * R. LAPACK-функция GEQP3.
FactorizationLQ	Выполняет LQ-разложение общей матрицы размером m на n: A = L * Q. Функция LAPACK GELQF.
FactorizationQL	Выполняет QL-разложение общей матрицы размером m на n: A = Q * L. Функция LAPACK GEQLF.
FactorizationRQ	Выполняет RQ-разложение общей матрицы размером m на n: A = R * Q. Функция LAPACK GERQF.

© 2000-2025, MetaQuotes Ltd.

FactorizationQR

Вычисляет QR-разложение общей матрицы размера m на n: A = Q * R. LAPACK-функция GEQRF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationQR(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrix&         Q,             // ортогональная матрица Q
   matrix&         R              // верхнетреугольная матрица R
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationQR(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixf&        Q,             // ортогональная матрица Q
   matrixf&        R              // верхнетреугольная матрица R
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationQR(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixc&        Q,             // унитарная матрица Q
   matrixc&        R              // верхнетреугольная матрица R
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationQR(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixcf&       Q,             // унитарная матрица Q
   matrixcf&       R              // верхнетреугольная матрица R
   );
```

Параметры

reduced

[in] Режим вычислений. Если параметр reduced установлен в значение true, тогда матрицы Q и R вычисляются с размерностями (M,K), (K,N). Если параметр reduced установлен в значение false, выполняется полный расчет матриц Q, R с размерностями (M,M), (M,N).

Q

[out] Ортогональная или унитарная матрица Q.

R

[out] Верхнетреугольная матрица R.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Примечание

Если *reduced* = true

При m >= n матрица Q имеет размер m на n, матрица R — n на n.

При m < n матрица Q имеет размер m на m, матрица R — m на n.

Если параметр *reduced* = false, матрица Q имеет размер m на m, а матрица R — m на n.

© 2000-2025, MetaQuotes Ltd.

FactorizationQRNonNeg

Вычисляет QR-разложение общей матрицы размера m на n: A = Q * R, где R — верхнетреугольная матрица с неотрицательными элементами на диагонали. LATPACK-функция GEQRFP.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationQRNonNeg(
   bool            reduced,      // режим вычислений: укороченный или полный
   matrix&         Q,            // ортогональная матрица Q
   matrix&         R             // верхнетреугольная матрица R
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationQRNonNeg(
   bool            reduced,      // режим вычислений: укороченный или полный
   matrixf&        Q,            // ортогональная матрица Q
   matrixf&        R             // верхнетреугольная матрица R
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationQRNonNeg(
   bool            reduced,      // режим вычислений: укороченный или полный
   matrixc&        Q,            // унитарная матрица Q
   matrixc&        R             // верхнетреугольная матрица R
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationQRNonNeg(
   bool            reduced,      // режим вычислений: укороченный или полный
   matrixcf&       Q,            // унитарная матрица Q
   matrixcf&       R             // верхнетреугольная матрица R
   );
```

Параметры

reduced

[in] Режим вычислений. Если параметр reduced установлен в значение true, тогда матрицы Q и R вычисляются с размерностями (M,K), (K,N). Если параметр reduced установлен в значение false, выполняется полный расчет матриц Q, R с размерностями (M,M), (M,N).

Q

[out] Ортогональная или унитарная матрица Q.

R

[out] Верхнетреугольная матрица R.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Если *reduced* = true

При m >= n матрица Q имеет размер m на n, матрица R — n на n.

При m < n матрица Q имеет размер m на m, матрица R — m на n.

Если параметр *reduced* = false, матрица Q имеет размер m на m, а матрица R — m на n.

© 2000-2025, MetaQuotes Ltd.

FactorizationQRPivot

Вычисляет QR-разложение общей матрицы размера m на n с перестановкой столбцов: A * P = Q * R. LAPACK-функция GEQP3.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationQRPivot(
   bool            reduced,      // режим вычислений: укороченный или полный
   long[]&         jpvt,         // массив с предопределенной перестановкой столбцов
   matrix&         Q,            // ортогональная матрица Q
   matrix&         R,            // верхнетреугольная матрица R
   matrix&         P             // матрица перестановок
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationQRPivot(
   bool            reduced,      // режим вычислений: укороченный или полный
   long[]&         jpvt,         // массив с предопределенной перестановкой столбцов
   matrixf&        Q,            // ортогональная матрица Q
   matrixf&        R,            // верхнетреугольная матрица R
   matrixf&        P             // матрица перестановок
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationQRPivot(
   bool            reduced,      // режим вычислений: укороченный или полный
   long[]&         jpvt,         // массив с предопределенной перестановкой столбцов
   matrixc&        Q,            // унитарная матрица Q
   matrixc&        R,            // верхнетреугольная матрица R
   matrixc&        P             // матрица перестановок
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationQRPivot(
   bool            reduced,      // режим вычислений: укороченный или полный
   long[]&         jpvt,         // массив с предопределенной перестановкой столбцов
   matrixcf&       Q,            // унитарная матрица Q
   matrixcf&       R,            // верхнетреугольная матрица R
   matrixcf&       P             // матрица перестановок
   );
```

Параметры

reduced

[in] Режим вычислений. Если параметр reduced установлен в значение true, тогда матрицы Q и R вычисляются с размерностями (M,K), (K,N). Если параметр reduced установлен в значение false, выполняется полный расчет матриц Q, R с размерностями (M,M), (M,N).

© 2000-2025, MetaQuotes Ltd.

jpvt

[in] Целочисленный массив размерности n. Если jpvt(i) ≠ 0, то i-й столбец матрицы A перемещается в начало матрицы AP до начала вычислений и фиксируется на этом месте в ходе разложения. Если jpvt(i) = 0, то i-й столбец матрицы A считается свободным (т.е. он может быть переставлен с любым другим свободным столбцом во время вычислений). Если массив jpvt имеет нулевой размер (или не инициализирован), предполагается, что все столбцы матрицы A свободны.

Q

[out] Ортогональная или унитарная матрица Q.

R

[out] Верхнетреугольная матрица R.

P

[out] Матрица перестановок P размером n на n.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Если *reduced* = true

При m >= n матрица Q имеет размер m на n, матрица R — n на n.

При m < n матрица Q имеет размер m на m, матрица R — m на n.

Если параметр *reduced* = false, матрица Q имеет размер m на m, а матрица R — m на n.

© 2000-2025, MetaQuotes Ltd.

FactorizationLQ

Выполняет LQ-разложение общей матрицы размером m на n: A = L * Q. Функция LAPACK GELQF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationLQ(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrix&         L,             // нижнетреугольная матрица L
   matrix&         Q              // ортогональная матрица Q
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationLQ(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixf&        L,             // нижнетреугольная матрица L
   matrixf&        Q              // ортогональная матрица Q
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationLQ(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixc&        L,             // нижнетреугольная матрица L
   matrixc&        Q              // унитарная матрица Q
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationLQ(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixcf&       L,             // нижнетреугольная матрица L
   matrixcf&       Q              // унитарная матрица Q
   );
```

Параметры

reduced

[in] Режим вычислений. Если параметр reduced установлен в значение true, тогда матрицы L и Q вычисляются с размерностями (M,K), (K,N). Если параметр reduced установлен в значение false, выполняется полный расчет матриц L и Q с размерностями (M,N), (N,N).

L

[out] Нижнетреугольная матрица L.

Q

[out] Ортогональная или унитарная матрица Q.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Примечание

Если *reduced* = true

При m <= n матрица L имеет размер m на m, матрица Q — m на n.

При m > n матрица L имеет размер m на n, матрица Q — n на n.

Если параметр *reduced* = false, матрица L имеет размер m на n, а матрица Q — n на n.

© 2000-2025, MetaQuotes Ltd.

FactorizationQL

Выполняет QL-разложение общей матрицы размером m на n: A = Q * L. Функция LAPACK GEQLF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationQL(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrix&         Q,             // ортогональная матрица Q
   matrix&         L              // нижнетреугольная матрица L
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationQL(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixf&        Q,             // ортогональная матрица Q
   matrixf&        L              // нижнетреугольная матрица L
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationQL(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixc&        Q,             // унитарная матрица Q
   matrixc&        L              // нижнетреугольная матрица L
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationQL(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixcf&       Q,             // унитарная матрица Q
   matrixcf&       L              // нижнетреугольная матрица L
   );
```

Параметры

reduced

[in] Режим вычислений. Если параметр reduced установлен в значение true, тогда матрицы Q и L вычисляются с размерностями (M,K), (K,N). Если параметр reduced установлен в значение false, выполняется полный расчет матриц Q и L с размерностями (M,M), (M,N).

Q

[out] Ортогональная или унитарная матрица Q.

L

[out] Нижнетреугольная матрица L.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Примечание

Если *reduced* = true

При m >= n матрица Q имеет размер m на n, матрица L — n на n.

При m < n матрица Q имеет размер m на m, матрица L — m на n.

Если параметр *reduced* = false, матрица Q имеет размер m на m, а матрица L — m на n.

© 2000-2025, MetaQuotes Ltd.

FactorizationRQ

Выполняет RQ-разложение общей матрицы размером m на n: A = R * Q. Функция LAPACK GERQF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationRQ(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrix&         R,             // верхнетреугольная матрица R
   matrix&         Q              // ортогональная матрица Q
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationRQ(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixf&        R,             // верхнетреугольная матрица R
   matrixf&        Q              // ортогональная матрица Q
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationRQ(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixc&        R,             // верхнетреугольная матрица R
   matrixc&        Q              // унитарная матрица Q
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationRQ(
   bool            reduced,       // режим вычислений: укороченный или полный
   matrixcf&       R,             // верхнетреугольная матрица R
   matrixcf&       Q              // унитарная матрица Q
   );
```

Параметры

reduced

[in] Режим вычислений. Если параметр reduced установлен в значение true, тогда матрицы R и Q вычисляются с размерностями (M,K), (K,N). Если параметр reduced установлен в значение false, выполняется полный расчет матриц R и Q с размерностями (M,N), (N,N).

R

[out] Верхнетреугольная матрица R.

Q

[out] Ортогональная или унитарная матрица Q.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Примечание

Если *reduced* = true

При m <= n матрица R имеет размер m на m, матрица Q — m на n.

При m > n матрица R имеет размер m на n, матрица Q — n на n.

Если параметр *reduced* = false, матрица R имеет размер m на n, а матрица Q — n на n.

© 2000-2025, MetaQuotes Ltd.

Linear Equations

Раздел предназначен для решения систем линейных уравнений вида A·X = B, где A — квадратная матрица коэффициентов, B — правая часть, а X — искомая матрица или вектор решений. Методы поддерживают различные типы данных: double, float, complex, complexf, и предоставляют реализацию на основе LAPACK-функций, обеспечивая высокую точность и производительность.

Функции разделены по типу матрицы A. Каждая функция имеет перегрузки для матричного и векторного представления правой части (B) и результата (X).

Функция	Выполняемое действие
LinearEquationsSolution	Вычисляет систему линейных уравнений с квадратной матрицей коэффициентов A и несколькими правыми частями. Имеет вид A*X = B, где A — общая квадратная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK GESV.
LinearEquationsSolutionTriangular	Вычисляет систему линейных уравнений с квадратно-треугольной матрицей коэффициентов A и несколькими правыми частями. Имеет вид A*X = B, где A — общая квадратная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK TRTRS.
LinearEquationsSolutionSy	Вычисляет систему линейных уравнений с симметричной или эрмитово сопряженной матрицей A и несколькими правыми частями. Имеет вид A*X = B, где A — симметричная или эрмитово сопряженная матрица размером n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функции из библиотеки LAPACK: SYSV и HESV.
LinearEquationsSolutionComplexSy	Вычисляет систему линейных уравнений с комплексной симметричной (не эрмитово сопряженной!) матрицей A и несколькими правыми частями. Имеет вид A*X = B, где A — комплексная симметричная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK SYSV.
LinearEquationsSolutionSyPD	Вычисляет систему линейных уравнений с симметричной или эрмитово сопряженной

© 2000-2025, MetaQuotes Ltd.

Функция	Выполняемое действие
	положительно определенной матрицей A и несколькими правыми частями. Система вида A*X = B, где A — симметричная или эрмитово сопряженная положительно определенная матрица размером n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK POSV.
LinearEquationsSolutionGeTri	Вычисляет систему линейных уравнений с общей (несимметричной) трехдиагональной матрицей коэффициентов A и несколькими правыми частями. Имеет вид A*X = B, где A — общая трехдиагональная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK GTSV.
LinearEquationsSolutionSyTriPD	Вычисляет систему линейных уравнений с симметричной трехдиагональной положительно определенной матрицей A и несколькими правыми частями. Имеет вид A*X = B, где A — симметричная трехдиагональная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK PTSV.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolution

Вычисляет систему линейных уравнений с квадратной матрицей коэффициентов A и несколькими правыми частями. Имеет вид A*X = B, где A — общая квадратная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK GESV.

Вычисления для типа matrix<double>

```
bool  matrix::LinearEquationsSolution(
   matrix&          B,             // матрица правых частей B
   matrix&          X              // матрица решений X
   );

bool  matrix::LinearEquationsSolution(
   vector&          B,             // матрица правых частей B
   vector&          X              // матрица решений X
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::LinearEquationsSolution(
   matrixf&         B,             // матрица правых частей B
   matrixf&         X              // матрица решений X
   );

bool  matrix::LinearEquationsSolution(
   vectorf&          B,             // матрица правых частей B
   vectorf&          X              // матрица решений X
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::LinearEquationsSolution(
   matrixc&         B,             // матрица правых частей B
   matrixc&         X              // матрица решений X
   );

bool  matrix::LinearEquationsSolution(
   vectorc&         B,             // матрица правых частей B
   vectorc&         X              // матрица решений X
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::LinearEquationsSolution(
   matrixcf&        B,             // матрица правых частей B
   matrixcf&        X              // матрица решений X
   );

bool  matrix::LinearEquationsSolution(
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorcf&       B,              // матрица правых частей B
   vectorcf&       X               // матрица решений X
   );
```

Параметры

B

[in] Матрица B, столбцы которой представляют собой правые части систем линейных уравнений. Если это вектор B, здесь содержится одна правая часть.

X

[out] Матрица или вектор X с решениями системы линейных уравнений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Выходная матрица X имеет те же размеры, что и входная матрица B. Выходной вектор X имеет ту же размерность, что и входной вектор B.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionTriangular

Вычисляет систему линейных уравнений с квадратно-треугольной матрицей коэффициентов A и несколькими правыми частями. Имеет вид A*X = B, где A — общая квадратная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK TRTRS.

Вычисления для типа matrix<double>

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // форма системы уравнений
   matrix&             B,            // матрица правых частей B
   matrix&             X             // матрица решений X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // форма системы уравнений
   vector&             B,            // матрица правых частей B
   vector&             X             // вектор решений X
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // форма системы уравнений
   matrixf&            B,            // матрица правых частей B
   matrixf&            X             // матрица решений X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // форма системы уравнений
   vectorf&            B,            // матрица правых частей B
   vectorf&            X             // вектор решений X
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // форма системы уравнений
   matrixc&            B,            // матрица правых частей B
   matrixc&            X             // матрица решений X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // форма системы уравнений
   vectorc&            B,            // матрица правых частей B
   vectorc&            X             // вектор решений X
   );
```

Вычисления для типа matrix<complexf>

© 2000-2025, MetaQuotes Ltd.

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,         // форма системы уравнений
   matrixcf&           B,             // матрица правых частей B
   matrixcf&           X              // матрица решений X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,         // форма системы уравнений
   vectorcf&           B,             // матрица правых частей B
   vectorcf&           X              // вектор решений X
   );
```

Параметры

trans

[in] Значение перечисления ENUM_EQUATIONS_FORM, которое определяет форму системы уравнений.

B

[in] Матрица B, столбцы которой представляют собой правые части систем линейных уравнений. Если это вектор B, здесь содержится одна правая часть.

X

[out] Матрица или вектор X с решениями системы линейных уравнений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Выходная матрица X имеет те же размеры, что и входная матрица B. Выходной вектор X имеет ту же размерность, что и входной вектор B.

ENUM_EQUATIONS_FORM

Перечисление, определяющее форму вычисляемой системы уравнений.

Идентификатор	Описание
EQUATIONSFORM_N	'N': A * X = B (без транспонирования)
EQUATIONSFORM_T	'T': A**T * X = B (транспонированная матрица A)
EQUATIONSFORM_C	'C': A**H * X = B (сопряженно-транспонированная матрица A)

Для вещественных матриц значение EQUATIONSFORM_C интерпретируется как EQUATIONSFORM_T.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionSy

Вычисляет систему линейных уравнений с симметричной или эрмитово сопряженной матрицей A и несколькими правыми частями. Имеет вид A*X = B, где A — симметричная или эрмитово сопряженная матрица размером n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функции из библиотеки LAPACK: SYSV и HESV.

Вычисления для типа matrix<double>

```
bool  matrix::LinearEquationsSolutionSy(
   matrix&          B,              // матрица правых частей B
   matrix&          X               // матрица решений X
   );

bool  matrix::LinearEquationsSolutionSy(
   vector&          B,              // матрица правых частей B
   vector&          X               // матрица решений X
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::LinearEquationsSolutionSy(
   matrixf&         B,              // матрица правых частей B
   matrixf&         X               // матрица решений X
   );

bool  matrix::LinearEquationsSolutionSy(
   vectorf&          B,              // матрица правых частей B
   vectorf&          X               // матрица решений X
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::LinearEquationsSolutionSy(
   matrixc&         B,              // матрица правых частей B
   matrixc&         X               // матрица решений X
   );

bool  matrix::LinearEquationsSolutionSy(
   vectorc&         B,              // матрица правых частей B
   vectorc&         X               // матрица решений X
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::LinearEquationsSolutionSy(
   matrixcf&        B,              // матрица правых частей B
   matrixcf&        X               // матрица решений X
   );
```

© 2000-2025, MetaQuotes Ltd.

```
bool  matrix::LinearEquationsSolutionSy(
   vectorcf&       B,             // матрица правых частей B
   vectorcf&       X              // матрица решений X
   );
```

Параметры

B

[in] Матрица B, столбцы которой представляют собой правые части систем линейных уравнений. Если это вектор B, здесь содержится одна правая часть.

X

[out] Матрица или вектор X с решениями системы линейных уравнений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Выходная матрица X имеет те же размеры, что и входная матрица B. Выходной вектор X имеет ту же размерность, что и входной вектор B.

На вход можно подать симметричную (эрмитово сопряженную), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionComplexSy

Вычисляет систему линейных уравнений с комплексной симметричной (не эрмитово сопряженной!) матрицей A и несколькими правыми частями. Имеет вид A*X = B, где A — комплексная симметричная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK SYSV.

Вычисления для типа matrix<complex>

```
bool  matrix::LinearEquationsSolutionComplexSy(
   matrixc&        B,             // матрица правых частей B
   matrixc&        X              // матрица решений X
   );

bool  matrix::LinearEquationsSolutionComplexSy(
   vectorc&        B,             // матрица правых частей B
   vectorc&        X              // матрица решений X
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::LinearEquationsSolutionComplexSy(
   matrixcf&       B,             // матрица правых частей B
   matrixcf&       X              // матрица решений X
   );

bool  matrix::LinearEquationsSolutionComplexSy(
   vectorcf&       B,             // матрица правых частей B
   vectorcf&       X              // матрица решений X
   );
```

Параметры

B

[in] Матрица B, столбцы которой представляют собой правые части систем линейных уравнений. Если это вектор B, здесь содержится одна правая часть.

X

[out] Матрица или вектор X с решениями системы линейных уравнений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Выходная матрица X имеет те же размеры, что и входная матрица B. Выходной вектор X имеет ту же размерность, что и входной вектор B.

На вход можно подать комплексную симметричную, верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются комплексными симметричными.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionSyPD

Вычисляет систему линейных уравнений с симметричной или эрмитово сопряженной положительно определенной матрицей A и несколькими правыми частями. Система вида A*X = B, где A — симметричная или эрмитово сопряженная положительно определенная матрица размером n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK POSV.

Вычисления для типа matrix<double>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrix&          B,              // матрица правых частей B
   matrix&          X               // матрица решений X
   );

bool  matrix::LinearEquationsSolutionSyPD(
   vector&          B,              // матрица правых частей B
   vector&          X               // матрица решений X
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrixf&         B,              // матрица правых частей B
   matrixf&         X               // матрица решений X
   );

bool  matrix::LinearEquationsSolutionSyPD(
   vectorf&          B,              // матрица правых частей B
   vectorf&          X               // матрица решений X
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrixc&         B,              // матрица правых частей B
   matrixc&         X               // матрица решений X
   );

bool  matrix::LinearEquationsSolutionSyPD(
   vectorc&         B,              // матрица правых частей B
   vectorc&         X               // матрица решений X
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrixcf&        B,              // матрица правых частей B
   matrixcf&        X               // матрица решений X
   );
```

© 2000-2025, MetaQuotes Ltd.

```
bool  matrix::LinearEquationsSolutionSyPD(
   vectorcf&       B,             // матрица правых частей B
   vectorcf&       X              // матрица решений X
   );
```

Параметры

B

[in] Матрица B, столбцы которой представляют собой правые части систем линейных уравнений. Если это вектор B, здесь содержится одна правая часть.

X

[out] Матрица или вектор X с решениями системы линейных уравнений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Выходная матрица X имеет те же размеры, что и входная матрица B. Выходной вектор X имеет ту же размерность, что и входной вектор B.

На вход можно подать симметричную (эрмитово сопряженную), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряжёнными).

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionGeTri

Вычисляет систему линейных уравнений с общей (несимметричной) трехдиагональной матрицей коэффициентов A и несколькими правыми частями. Имеет вид A*X = B, где A — общая трехдиагональная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK GTSV.

Вычисления для типа matrix<double>

```
bool  matrix::LinearEquationsSolutionGeTri(
   matrix&          B,             // матрица правых частей B
   matrix&          X              // матрица решений X
   );

bool  matrix::LinearEquationsSolutionGeTri(
   vector&          B,             // матрица правых частей B
   vector&          X              // матрица решений X
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::LinearEquationsSolutionGeTri(
   matrixf&         B,             // матрица правых частей B
   matrixf&         X              // матрица решений X
   );

bool  matrix::LinearEquationsSolutionGeTri(
   vectorf&          B,             // матрица правых частей B
   vectorf&          X              // матрица решений X
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::LinearEquationsSolutionGeTri(
   matrixc&         B,             // матрица правых частей B
   matrixc&         X              // матрица решений X
   );

bool  matrix::LinearEquationsSolutionGeTri(
   vectorc&         B,             // матрица правых частей B
   vectorc&         X              // матрица решений X
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::LinearEquationsSolutionGeTri(
   matrixcf&        B,             // матрица правых частей B
   matrixcf&        X              // матрица решений X
   );

bool  matrix::LinearEquationsSolutionGeTri(
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorcf&       B,              // матрица правых частей B
   vectorcf&       X               // матрица решений X
   );
```

Параметры

B

[in] Матрица B, столбцы которой представляют собой правые части систем линейных уравнений. Если это вектор B, здесь содержится одна правая часть.

X

[out] Матрица или вектор X с решениями системы линейных уравнений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Выходная матрица X имеет те же размеры, что и входная матрица B. Выходной вектор X имеет ту же размерность, что и входной вектор B.

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionSyTriPD

Вычисляет систему линейных уравнений с симметричной трехдиагональной положительно определенной матрицей A и несколькими правыми частями. Имеет вид A*X = B, где A — симметричная трехдиагональная матрица размера n на n, столбцы матрицы B представляют собой отдельные правые части, а столбцы матрицы X содержат соответствующие решения. Функция LAPACK PTSV.

Вычисления для типа matrix<double>

```
bool  matrix::LinearEquationsSolutionSyTriPD(
   matrix&          B,              // матрица правых частей B
   matrix&          X               // матрица решений X
   );

bool  matrix::LinearEquationsSolutionSyTriPD(
   vector&          B,              // матрица правых частей B
   vector&          X               // матрица решений X
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::LinearEquationsSolutionSyTriPD(
   matrixf&         B,              // матрица правых частей B
   matrixf&         X               // матрица решений X
   );

bool  matrix::LinearEquationsSolutionSyTriPD(
   vectorf&          B,              // матрица правых частей B
   vectorf&          X               // матрица решений X
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::LinearEquationsSolutionSyTriPD(
   matrixc&         B,              // матрица правых частей B
   matrixc&         X               // матрица решений X
   );

bool  matrix::LinearEquationsSolutionSyTriPD(
   vectorc&         B,              // матрица правых частей B
   vectorc&         X               // матрица решений X
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::LinearEquationsSolutionSyTriPD(
   matrixcf&        B,              // матрица правых частей B
   matrixcf&        X               // матрица решений X
   );
```

© 2000-2025, MetaQuotes Ltd.

```
bool  matrix::LinearEquationsSolutionSyTriPD(
   vectorcf&       B,             // матрица правых частей B
   vectorcf&       X              // матрица решений X
   );
```

Параметры

B

[in] Матрица B, столбцы которой представляют собой правые части систем линейных уравнений. Если это вектор B, здесь содержится одна правая часть.

X

[out] Матрица или вектор X с решениями системы линейных уравнений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Выходная матрица X имеет те же размеры, что и входная матрица B. Выходной вектор X имеет ту же размерность, что и входной вектор B.

© 2000-2025, MetaQuotes Ltd.

Factorizations

Раздел Factorizations содержит функции для выполнения различных видов матричных разложений, используемых при численном решении систем линейных уравнений, анализе устойчивости и других задачах линейной алгебры. Эти разложения преобразуют исходную матрицу в более простые формы, что облегчает дальнейшие вычисления. Все функции реализованы с использованием LAPACK-подпрограмм и поддерживают типы double, float, complex и complexf.

Функции этого раздела применяются:

- Для предварительной обработки матриц при решении систем линейных уравнений;
- При расчете определителей, ранга и инверсии матриц;
- Для оценки устойчивости численных методов;
- При решении задач в спектральной теории и методах оптимизации.

Разложение матриц — ключевой этап во многих алгоритмах линейной алгебры, и данный раздел обеспечивает доступ к наиболее эффективным и проверенным методам факторизации.

Функция	Выполняемое действие
FactorizationPLU	Вычисляет LU-разложение общей матрицы A размером m на n с использованием частичного выбора ведущего элемента и перестановок строк. Разложение имеет вид A = P * L * U. LAPACK-функция GETRF.
FactorizationPLUGeTrid	Вычисляет LU-разложение общей (несимметричной) трехдиагональной матрицы A размером n на n с использованием частичного выбора ведущего элемента и перестановок строк. Разложение имеет вид A = P * L * U. LAPACK-функция GTTRF.
FactorizationLDL	Вычисляет разложение для вещественной симметричной или комплексной эрмитовой матрицы A с использованием диагонального выбора ведущего элемента по методу Банча-Кауфмана. LAPACK-функции SYTRF и HETRF.
FactorizationLDLSyTridPD	Вычисляет разложение симметричной положительно определенной (для вещественных данных) или эрмитовой положительно определенной (для комплексных данных) трехдиагональной матрицы A. LAPACK-функция PTTRF.
FactorizationCholesky	Вычисляет разложение для вещественной симметричной или комплексной эрмитовой положительно-определенной матрицы A. LAPACK-функция POTRF.
FactorizationCholeskySyPS	Вычисляет разложение Холецкого с полным выбором ведущего элемента (complete pivoting)

© 2000-2025, MetaQuotes Ltd.

Функция	Выполняемое действие
	для вещественной симметричной (или комплексной эрмитовой) положительно полуопределенной матрицы A размером n на n. Функция LAPACK PSTRF.

© 2000-2025, MetaQuotes Ltd.

FactorizationPLU

Вычисляет LU-разложение общей матрицы A размером m на n с использованием частичного выбора ведущего элемента и перестановок строк. Разложение имеет вид:

A = P * L * U

где P — матрица перестановок, L — нижнетреугольная матрица с единицами на диагонали (или нижняя трапециевидная матрица, если m > n), а U — верхнетреугольная матрица (или верхняя трапециевидная матрица, если m < n). LAPACK-функция GETRF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationPLU(
   matrix&          P,             // матрица перестановок P
   matrix&          L,             // нижнетреугольная матрица L
   matrix&          U              // верхнетреугольная матрица U
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationPLU(
   matrixf&         P,             // матрица перестановок P
   matrixf&         L,             // нижнетреугольная матрица L
   matrixf&         U              // верхнетреугольная матрица U
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationPLU(
   matrixc&         P,             // матрица перестановок P
   matrixc&         L,             // нижнетреугольная матрица L
   matrixc&         U              // верхнетреугольная матрица U
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationPLU(
   matrixcf&        P,             // матрица перестановок P
   matrixcf&        L,             // нижнетреугольная матрица L
   matrixcf&        U              // верхнетреугольная матрица U
   );
```

Параметры

P

[out] Матрица перестановок P.

L

[out] Нижнетреугольная матрица с единицами по диагонали.

U

[out] Верхнетреугольная матрица.

© 2000-2025, MetaQuotes Ltd.

Возвращаемое значение

Возвращает true в случае успеха и false в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

FactorizationPLUGeTrid

Вычисляет LU-разложение общей (несимметричной) трехдиагональной матрицы A размером m на m с использованием частичного выбора ведущего элемента и перестановок строк. Разложение имеет вид:

A = P * L * U

где P — матрица перестановок, L — нижнетреугольная матрица с единицами по диагонали, а U — верхнетреугольная. LAPACK-функция GTTRF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationPLUGeTrid(
   matrix&          P,             // матрица перестановок P
   matrix&          L,             // нижнетреугольная матрица L
   matrix&          U              // верхнетреугольная матрица U
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationPLUGeTrid(
   matrixf&         P,             // матрица перестановок P
   matrixf&         L,             // нижнетреугольная матрица L
   matrixf&         U              // верхнетреугольная матрица U
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationPLUGeTrid(
   matrixc&         P,             // матрица перестановок P
   matrixc&         L,             // нижнетреугольная матрица L
   matrixc&         U              // верхнетреугольная матрица U
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationPLUGeTrid(
   matrixcf&        P,             // матрица перестановок P
   matrixcf&        L,             // нижнетреугольная матрица L
   matrixcf&        U              // верхнетреугольная матрица U
   );
```

Параметры

P

[out] Матрица перестановок P.

L

[out] Нижнетреугольная матрица с единицами по диагонали.

U

[out] Верхнетреугольная матрица.

© 2000-2025, MetaQuotes Ltd.

Возвращаемое значение

Возвращает true в случае успеха и false в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

FactorizationLDL

Вычисляет разложение для вещественной симметричной или комплексной эрмитовой матрицы A с использованием диагонального выбора ведущего элемента по методу Банча-Кауфмана. Разложение имеет вид:

A = L * D * L**T, если используется нижнетреугольная (или симметричная) матрица A,

или

A = U**T * D * U, если используется верхнетреугольная матрица A,

где L — нижнетреугольная матрица с единицами на диагонали, U — верхнетреугольная матрица с единицами на диагонали. D — симметричная блочно-диагональная матрица с диагональными блоками размером 1x1 и 2x2. LAPACK-функции SYTRF и HETRF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationLDL(
   matrix&         L,             // нижнетреугольная или верхнетреугольная матрица
   matrix&         D              // диагональная матрица D
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationLDL(
   matrixf&        L,             // нижнетреугольная или верхнетреугольная матрица
   matrixf&        D              // диагональная матрица D
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationLDL(
   matrixc&        L,             // нижнетреугольная или верхнетреугольная матрица
   matrixc&        D              // диагональная матрица D
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationLDL(
   matrixcf&       L,             // нижнетреугольная или верхнетреугольная матрица
   matrixcf&       D              // диагональная матрица D
   );
```

Параметры

L

[out] Нижнетреугольная или верхнетреугольная матрица с единицами по диагонали.

D

[out] Симметричная блочно-диагональная матрица D.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успеха и false в случае ошибки.

Примечание

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряженными).

© 2000-2025, MetaQuotes Ltd.

FactorizationLDLSyTridPD

Вычисляет разложение (факторизацию) симметричной положительно определенной (для вещественных данных) или эрмитовой положительно определенной (для комплексных данных) трехдиагональной матрицы A. Разложение имеет вид:

A = L * D * L**T, если используется нижнетреугольная (или симметричная) матрица A,

или

A = U**T * D * U, если используется верхнетреугольная матрица A,

где L — нижнетреугольная матрица с единицами на диагонали, U — верхнетреугольная матрица с единицами на диагонали. D — диагональная матрица. LAPACK-функция PTTRF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrix&         L,             // нижнетреугольная или верхнетреугольная матрица
   matrix&         D              // диагональная матрица D
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrixf&        L,             // нижнетреугольная или верхнетреугольная матрица
   matrixf&        D              // диагональная матрица D
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrixc&        L,             // нижнетреугольная или верхнетреугольная матрица
   matrixc&        D              // диагональная матрица D
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrixcf&       L,             // нижнетреугольная или верхнетреугольная матрица
   matrixcf&       D              // диагональная матрица D
   );
```

Параметры

L

[out] Нижнетреугольная или верхнетреугольная матрица с единицами по диагонали.

D

[out] Диагональная матрица D.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успеха и false в случае ошибки.

Примечание

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряженными).

© 2000-2025, MetaQuotes Ltd.

FactorizationCholesky

Вычисляет разложение для вещественной симметричной или комплексной эрмитовой положительно-определенной матрицы A. Разложение имеет вид:

A = L * L**T, если используется нижнетреугольная (или симметричная) матрица A,

или

A = U**T * U, если используется верхнетреугольная матрица A,

где L — нижнетреугольная матрица, U — верхнетреугольная. LAPACK-функция POTRF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationCholesky(
   matrix&         L             // нижнетреугольная или верхнетреугольная матрица
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationCholesky(
   matrixf&        L             // нижнетреугольная или верхнетреугольная матрица
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationCholesky(
   matrixc&        L             // нижнетреугольная или верхнетреугольная матрица
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationCholesky(
   matrixcf&       L             // нижнетреугольная или верхнетреугольная матрица
   );
```

Параметры

L

[out] Нижнетреугольная или верхнетреугольная матрица.

Возвращаемое значение

Возвращает true в случае успеха и false в случае ошибки.

Примечание

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряженными).

© 2000-2025, MetaQuotes Ltd.

FactorizationCholeskySyPS

Вычисляет разложение Холецкого с полным выбором ведущего элемента (complete pivoting) для вещественной симметричной (или комплексной эрмитовой) положительно полуопределенной матрицы A размером n на n. Разложение имеет вид:

P**T * A * P = L * L**T, если используется нижнетреугольная (или симметричная) матрица A,

или

P**T * A * P = U**T * U, если используется верхнетреугольная матрица A,

где P — матрица перестановок, L — нижнетреугольная матрица, а U — верхнетреугольная. Функция LAPACK PSTRF.

Вычисления для типа matrix<double>

```
bool  matrix::FactorizationCholeskySyPS(
   double          tol           // допуск
   matrix&         P,            // матрица перестановок P
   matrix&         L             // нижнетреугольная или верхнетреугольная матрица
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::FactorizationCholeskySyPS(
   float           tol           // допуск
   matrixf&        P,            // матрица перестановок P
   matrixf&        L             // нижнетреугольная или верхнетреугольная матрица
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::FactorizationCholeskySyPS(
   double          tol           // допуск
   matrixc&        P,            // матрица перестановок P
   matrixc&        L             // нижнетреугольная или верхнетреугольная матрица
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::FactorizationCholeskySyPS(
   float           tol           // допуск
   matrixcf&       P,            // матрица перестановок P
   matrixcf&       L             // нижнетреугольная или верхнетреугольная матрица
   );
```

Параметры

tol

[in] Допуск, устанавливаемый пользователем. Если tol < 0, тогда используется значение n*ε*max(A[k,k]), где ε — машинная точность. Алгоритм завершается на шаге k-1, если ведущий элемент (pivot) <= tol.

© 2000-2025, MetaQuotes Ltd.

P

[out] Матрица перестановок P.

L

[out] Нижнетреугольная или верхнетреугольная матрица.

Возвращаемое значение

Возвращает true в случае успеха и false в случае ошибки.

Примечание

На вход можно подать симметричную (эрмитову), верхнетреугольную или нижнетреугольную матрицу. Треугольные матрицы подразумеваются симметричными (эрмитово-сопряженными).

© 2000-2025, MetaQuotes Ltd.

Dynamic Mode Decomposition

Функции разложения по динамическим модам (Dynamic Mode Decomposition, DMD) предназначены для вычисления оператора эволюции на основе срезов "snapshots", что позволяет эффективно анализировать линейную динамику в расширенных фазовых пространствах. Эти методы обеспечивают экономное использование памяти при представлении операторов эволюции, при этом DynamicModeDecompositionQR использует QR-разложение для сжатия данных, что повышает вычислительную эффективность.

Функция	Выполняемое действие
DynamicModeDecomposition	Вычисляет DMD для пары матриц срезов данных. Функция Lapack GEDMD.
DynamicModeDecompositionQR	Вычисляет DMD для пары матриц срезов данных с использованием QR-факторизации для сжатия данных. Функция Lapack GEDMDQ.

Метод разложения по динамическим модам (Dynamic mode decomposition, DMD) представляет собой алгоритм поиска оператора эволюции (решения обратной операторной задачи) в конечномерном пространстве решений задачи (численных или экспериментально полученных) по набору решений (срезов, "snapshots") в некоторые последовательные моменты времени. Расширение фазового пространства за счет использования нелинейного (относительно переменных задачи) базиса позволяет построить глобальный линейный оператор, описывающий линейную эволюцию в расширенном "спрямляющем пространстве" (оператор Купмана) и сопряженный ему оператор Перрона-Фробениуса. Метод DMD эквивалентен сжатому представлению линейного оператора эволюции в виде произведения прямоугольных матриц, что обеспечивает значительную экономию необходимой памяти при вычислениях.

© 2000-2025, MetaQuotes Ltd.

DynamicModeDecomposition

Вычисляет разложение по динамическим модам (DMD) для пары матриц срезов данных. Функция LAPACK GEDMD. Для входных матриц X и Y, таких что Y = A*X, где матрица A недоступна, функция вычисляет определенное число пар Ритца матрицы A, используя стандартное извлечение Рэлея-Ритца из подпространства range(X), которое определяется с использованием ведущих левых сингулярных векторов матрицы X. Дополнительно функция может возвращать остатки вычисленных пар Ритца, информацию для уточнения векторов Ритца, или собственные векторы точного DMD (Exact DMD). Обе матрицы должны быть одинакового размера.

Вычисления для типа matrix<double>

```
bool  matrix::DynamicModeDecomposition(
   matrix&              B,                  // вторая матрица среза данных в пар
   ENUM_DMD_SCALE       jobs,               // необходимость масштабировать нача
   ENUM_DMD_EIGV        jobz,               // необходимость вычисления собстве
   ENUM_DMD_RESIDUALS   jobr,               // необходимость вычисления остатко
   ENUM_DMD_REFINE      jobf,               // сохранять ли информацию, необход
   ENUM_SVD_ALG         whtsvd,             // выбор алгоритма SVD из библиотеки
   long                 nrnk,               // определяет режим вычисления числ
   double               tol,                // допустимая погрешность для усече
   vectorc&             eigen_values,       // вектор собственных значений (зна
   matrix&              left_vectors,       // матрица левых сингулярных векторо
   matrix&              Z                   // матрица векторов Ритца
   vector&              residuals,          // остатки для пар Ритца
   matrix&              res_vectors,        // остаточные векторы для пар Ритца
   matrix&              B,                  // вычисленные векторы
   matrix&              W,                  // собственные векторы матрицы Рэле
   matrix&              S                   // вычисленные векторы
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::DynamicModeDecomposition(
   matrixf&             B,                  // вторая матрица среза данных в пар
   ENUM_DMD_SCALE       jobs,               // необходимость масштабировать нача
   ENUM_DMD_EIGV        jobz,               // необходимость вычисления собстве
   ENUM_DMD_RESIDUALS   jobr,               // необходимость вычисления остатко
   ENUM_DMD_REFINE      jobf,               // сохранять ли информацию, необход
   ENUM_SVD_ALG         whtsvd,             // выбор алгоритма SVD из библиотеки
   long                 nrnk,               // определяет режим вычисления числ
   float                tol,                // допустимая погрешность для усече
   vectorcf&            eigen_values,       // вектор собственных значений (зна
   matrixf&             left_vectors,       // матрица левых сингулярных векторо
   matrixf&             Z                   // матрица векторов Ритца
   vectorf&             residuals,          // остатки для пар Ритца
   matrixf&             res_vectors,        // остаточные векторы для пар Ритца
   matrixf&             B,                  // вычисленные векторы
   matrixf&             W,                  // собственные векторы матрицы Рэле
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixf&              S                         // вычисленные векторы
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::DynamicModeDecomposition(
   matrixc&              B,                        // вторая матрица среза данных в пар
   ENUM_DMD_SCALE        jobs,                     // необходимость масштабировать нача
   ENUM_DMD_EIGV         jobz,                     // необходимость вычисления собствен
   ENUM_DMD_RESIDUALS    jobr,                     // необходимость вычисления остатков
   ENUM_DMD_REFINE       jobf,                     // сохранять ли информацию, необходи
   ENUM_SVD_ALG          whtsvd,                   // выбор алгоритма SVD из библиотеки
   long                  nrnk,                     // определяет режим вычисления числе
   double                tol,                      // допустимая погрешность для усечен
   vectorc&              eigen_values,             // вектор собственных значений (знач
   matrixc&              left_vectors,             // матрица левых сингулярных векторо
   matrixc&              Z                         // матрица векторов Ритца
   vectorc&              residuals,                // остатки для пар Ритца
   matrixc&              res_vectors,              // остаточные векторы для пар Ритца
   matrixc&              B,                        // вычисленные векторы
   matrixc&              W,                        // собственные векторы матрицы Рэлея
   matrixc&              S                         // вычисленные векторы
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::DynamicModeDecomposition(
   matrixcf&             B,                        // вторая матрица среза данных в пар
   ENUM_DMD_SCALE        jobs,                     // необходимость масштабировать нача
   ENUM_DMD_EIGV         jobz,                     // необходимость вычисления собствен
   ENUM_DMD_RESIDUALS    jobr,                     // необходимость вычисления остатков
   ENUM_DMD_REFINE       jobf,                     // сохранять ли информацию, необходи
   ENUM_SVD_ALG          whtsvd,                   // выбор алгоритма SVD из библиотеки
   long                  nrnk,                     // определяет режим вычисления числе
   float                 tol,                      // допустимая погрешность для усечен
   vectorcf&             eigen_values,             // вектор собственных значений (знач
   matrixcf&             left_vectors,             // матрица левых сингулярных векторо
   matrixcf&             Z                         // матрица векторов Ритца
   vectorcf&            residuals,                // остатки для пар Ритца
   matrixcf&             res_vectors,              // остаточные векторы для пар Ритца
   matrixcf&             B,                        // вычисленные векторы
   matrixcf&             W,                        // собственные векторы матрицы Рэлея
   matrixcf&             S                         // вычисленные векторы
   );
```

Параметры

Y

[in] Вторая матрица среза данных в паре.

© 2000-2025, MetaQuotes Ltd.

jobs

[in] Значение из перечисления ENUM_DMD_SCALE, которое определяет необходимость масштабирования исходных срезов с использованием диагональной матрицы.

jobz

[in] Значение из перечисления ENUM_DMD_EIGV, которое определяет необходимость вычисления собственных векторов (моды Купмана).

jobr

[in] Значение из перечисления ENUM_DMD_RESIDUALS, которое определяет, будут ли вычисляться остатки.

jobf

[in] Значение из перечисления ENUM_DMD_REFINE, которое определяет, следует ли сохранять информацию, необходимую для постобработки (например, вычисления уточненных векторов Ритца).

whtsvd

[in] Значение из перечисления ENUM_SVD_ALG, которое позволяет выбрать алгоритм SVD из библиотеки LAPACK.

nrnk

[in] Значение определяет способ вычисления числового ранга, т.е. способ усечения малых сингулярных значений входной матрицы X. При

NRNK = -1 :: i-e сингулярное значение sigma(i) усекается, если sigma(i) <= TOL*sigma(1). Этот вариант рекомендуется.

При NRNK = -2 :: i-e сингулярное значение sigma(i) усекается, если sigma(i) <= TOL*sigma(i-1). Эта опция включена для исследований и разработок. Требует высокой точности SVD, что не всегда достигаемо.

Числовой ранг может быть задан с помощью положительного значения NRNK следующим образом: 0 < NRNK <= N :: будут использоваться не более NRNK наибольших сингулярных значений. Если число вычисленных ненулевых сингулярных значений меньше NRNK, то будут использоваться только эти ненулевые значения, а фактически используемая размерность будет меньше NRNK.

tol

[in] Допустимая погрешность для усечения малых сингулярных значений. 0 <= TOL < 1

eigen_values

[out] Вектор собственных значений размера K. Ведущие K (K<=N) элементов в EIGS содержат вычисленные собственные значения (значения Ритца).

left_vectors

[out] Матрица левых сингулярных векторов. Ведущие K столбцов содержат базис POD (правильная ортогональная декомпозиция), т.е. ведущие K левых сингулярных векторов входной матрицы X. Все N столбцов содержат все левые сингулярные векторы матрицы X.

Z

[out] Матрица векторов Ритца. Если JOBZ =='V', то Z содержит векторы Ритца. Z(:,i) — собственный вектор i-го значения Ритца; ||Z(:,i)||_2=1. Если JOBZ == 'F', то векторы Z(:,i) задаются неявно как столбцы X(:,1:K)*W(1:K,1:K), т.е. X(:,1:K)*W(:,i) является собственным

© 2000-2025, MetaQuotes Ltd.

вектором, соответствующим EIGS(i). Столбцы W(1:k,1:K) представляют собой вычисленные собственные векторы отношения Рэлея размером K-на-K.

residuals

[out] Остатки для K вычисленных пар Ритца.

res_vectors

[out] Остаточные векторы для K вычисленных пар Ритца.

B

[out] Матрица размером M на K (K<=N).

Если JOBF =='R', B(1:M,1:K) содержит A*U(:,1:K) и может использоваться для уточненных векторов.

Если JOBF == 'E', B(1:M,1:K) содержит A*U(:,1:K)*W(1:K,1:K), которые являются векторами из точного DMD (Exact), с точностью до масштабирования по обратным собственным значениям.

Если JOBF =='N', то матрица B не используется.

W

[out] Матрица размером K на K (K<=N). Содержит K вычисленных собственных векторов матричного отношения Рэлея. Векторы Ритца (возвращаемые в Z) являются произведением X (содержащего базис POD для входной матрицы X) и W.

S

[out] Матрица размером K на K (K<=N). Массив S(1:K,1:K) используется для матричного отношения Рэлея. Содержимое перезаписывается во время разложения собственных значений с помощью GEEV.

Возвращаемое значение

Возвращает true в случае успеха или false, если произошла ошибка.

Примечание

Количество строк (M) матрицы не должно быть меньше количества столбцов (N).

ENUM_DMD_SCALE

Перечисление, которое определяет, масштабируются ли исходные срезы данных с помощью диагональной матрицы.

Идентификатор	Описание
DMDSCALE_S	'S' — матрицы срезов данных X и Y умножаются на диагональную матрицу D так, что X*D имеет единичные ненулевые столбцы.
DMDSCALE_C	'C' — срезы масштабируются так же, как и в варианте 'S'. Если обнаружено, что i-й столбец матрицы X является

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
	нулевым вектором, а соответствующий i-й столбец матрицы Y не равен нулю, то i-й столбец Y устанавливается равным нулю.
DMDSCALE_Y	'Y' — матрицы срезов данных X и Y умножаются на диагональную матрицу D так, что Y*D имеет единичные ненулевые столбцы.
DMDSCALE_N	'N' — масштабирование данных не предусмотрено.

ENUM_DMD_EIGV

Перечисление, которое определяет, будут ли вычисляться собственные векторы (моды Купмана).

Идентификатор	Описание
DMDEIGV_V	'V' — собственные векторы (моды Купмана) будут вычисляться.
DMDEIGV_F	'F' — будут возвращены собственные векторы (моды Купмана) в факторизованной форме.
DMDEIGV_N	'N' — собственные векторы не вычисляются

ENUM_DMD_RESIDUALS

Перечисление, которое определяет необходимость вычисления остатков.

Идентификатор	Описание
DMDRESIDUALS_R	'R' — вычислять остатки для собственных пар.
DMDRESIDUALS_N	'N' — не вычислять остатки.

ENUM_DMD_REFINE

Перечисление, которое определяет, следует ли хранить информацию, необходимую для постобработки.

Идентификатор	Описание
DMDREFINE_R	'R' — матрица, необходимая для уточнения векторов Ритца, вычисляется и сохраняется в массиве B.
DMDREFINE_E	'E' — немасштабированные собственные векторы точного DMD вычисляются и возвращаются в массиве B.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
DMDREFINE_N	'N' — данные для уточнения собственных векторов не вычисляются.

ENUM_SVD_ALG

Перечисление для выбора алгоритма SVD.

Идентификатор	Описание
SVDALG_1	1: GESVD (алгоритм QR SVD)
SVDALG_2	2: GESDD (алгоритм Divide and Conquer)
SVDALG_3	3: GESVDQ (предобусловленный алгоритм QR SVD; этот и следующий варианты самые точные)
SVDALG_4	4: GEJSV (предобусловленный алгоритм Jacobi SVD; этот и предыдущий варианты самые точные)

© 2000-2025, MetaQuotes Ltd.

DynamicModeDecompositionQR

Вычисляет разложение по динамическим модам (DMD) для пары матриц срезов данных с использованием QR-факторизации для сжатия данных. Функция LAPACK GEDMDQ. Для входных матриц X и Y, таких что Y = A*X, где матрица A недоступна, функция вычисляет определенное число пар Ритца матрицы A, используя стандартное извлечение Рэлея-Ритца из подпространства range(X), которое определяется с использованием ведущих левых сингулярных векторов матрицы X. Дополнительно функция может возвращать остатки вычисленных пар Ритца, информацию для уточнения векторов Ритца, или собственные векторы точного DMD (Exact DMD).

На вход подается матрица F размером M×N. Столбцы матрицы F представляют собой последовательность срезов данных (snapshots) с одной траектории, снятых через равные дискретные интервалы времени. Предполагается, что нормы столбцов матрицы F находятся в диапазоне нормализованных чисел с плавающей запятой.

Вычисления для типа matrix<double>

```
bool  matrix::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE        jobs,                 // нужно ли масштабировать начальные
   ENUM_DMDQ_EIGV        jobz,                 // нужно ли вычислять собственные ве
   ENUM_DMD_RESIDUALS    jobr,                 // нужно ли вычислять остатки
   ENUM_DMDQ_Q           jobq,                 // нужно ли явно вычислять и возвраш
   ENUM_DMDQ_R           jobt,                 // нужно ли возвращать верхнетреугол
   ENUM_DMD_REFINE       jobf,                 // сохранять ли информацию, необходи
   ENUM_SVD_ALG          whtsvd,               // выбор алгоритма SVD из библиотеки
   long                  nrnk,                 // определяет режим вычисления числе
   double                tol,                  // допустимая погрешность для усечен
   vectorc&              eigen_values,         // вектор собственных значений (знач
   matrix&               Q,                    // матрица Q из QR-разложения
   matrix&               R,                    // матрица R из QR-разложения
   matrix&               left_vectors,         // матрица левых сингулярных векторо
   matrix&               Z                     // матрица векторов Ритца
   vector&               residuals,            // остатки для пар Ритца
   matrix&               B,                    // вычисленные векторы
   matrix&               V,                    // собственные векторы матричного от
   matrix&               S                     // вычисленные векторы
   );
```

Вычисления для типа matrix<float>

```
bool  matrixf::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE        jobs,                 // нужно ли масштабировать начальные
   ENUM_DMDQ_EIGV        jobz,                 // нужно ли вычислять собственные ве
   ENUM_DMD_RESIDUALS    jobr,                 // нужно ли вычислять остатки
   ENUM_DMDQ_Q           jobq,                 // нужно ли явно вычислять и возвраш
   ENUM_DMDQ_R           jobt,                 // нужно ли возвращать верхнетреугол
   ENUM_DMD_REFINE       jobf,                 // сохранять ли информацию, необходи
   ENUM_SVD_ALG          whtsvd,               // выбор алгоритма SVD из библиотеки
   long                  nrnk,                 // определяет режим вычисления числе
```

© 2000-2025, MetaQuotes Ltd.

```
   float                tol,                // допустимая погрешность для усечен
   vectorcf&            eigen_values,       // вектор собственных значений (знач
   matrixf&             Q,                  // матрица Q из QR-разложения
   matrixf&             R,                  // матрица R из QR-разложения
   matrixf&             left_vectors,       // матрица левых сингулярных векторо
   matrixf&             Z                   // матрица векторов Ритца
   vectorf&             residuals,          // остатки для пар Ритца
   matrixf&             B,                  // вычисленные векторы
   matrixf&             V,                  // собственные векторы матричного от
   matrixf&             S                   // вычисленные векторы
   );
```

Вычисления для типа matrix<complex>

```
bool  matrixc::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE       jobs,               // нужно ли масштабировать начальные
   ENUM_DMDQ_EIGV       jobz,               // нужно ли вычислять собственные ве
   ENUM_DMD_RESIDUALS   jobr,               // нужно ли вычислять остатки
   ENUM_DMDQ_Q          jobq,               // нужно ли явно вычислять и возвраш
   ENUM_DMDQ_R          jobt,               // нужно ли возвращать верхнетреугол
   ENUM_DMD_REFINE      jobf,               // сохранять ли информацию, необходи
   ENUM_SVD_ALG         whtsvd,             // выбор алгоритма SVD из библиотеки
   long                 nrnk,               // определяет режим вычисления числе
   double               tol,                // допустимая погрешность для усечен
   vectorc&             eigen_values,       // вектор собственных значений (знач
   matrixc&             Q,                  // матрица Q из QR-разложения
   matrixc&             R,                  // матрица R из QR-разложения
   matrixc&             left_vectors,       // матрица левых сингулярных векторо
   matrixc&             Z                   // матрица векторов Ритца
   vector&              residuals,          // остатки для пар Ритца
   matrixc&             B,                  // вычисленные векторы
   matrixc&             V,                  // собственные векторы матричного от
   matrixc&             S                   // вычисленные векторы
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE       jobs,               // нужно ли масштабировать начальные
   ENUM_DMDQ_EIGV       jobz,               // нужно ли вычислять собственные ве
   ENUM_DMD_RESIDUALS   jobr,               // нужно ли вычислять остатки
   ENUM_DMDQ_Q          jobq,               // нужно ли явно вычислять и возвраш
   ENUM_DMDQ_R          jobt,               // нужно ли возвращать верхнетреугол
   ENUM_DMD_REFINE      jobf,               // сохранять ли информацию, необходи
   ENUM_SVD_ALG         whtsvd,             // выбор алгоритма SVD из библиотеки
   long                 nrnk,               // определяет режим вычисления числе
   float                tol,                // допустимая погрешность для усечен
   vectorcf&            eigen_values,       // вектор собственных значений (знач
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixcf&            Q,                  // матрица Q из QR-разложения
   matrixcf&            R,                  // матрица R из QR-разложения
   matrixcf&            left_vectors,       // матрица левых сингулярных векторо
   matrixcf&            Z                   // матрица векторов Ритца
   vectorf&             residuals,          // остатки для пар Ритца
   matrixcf&            B,                  // вычисленные векторы
   matrixcf&            V,                  // собственные векторы матричного от
   matrixcf&            S                   // вычисленные векторы
   );
```

Параметры

jobs

[in] Значение из перечисления ENUM_DMD_SCALE, которое определяет необходимость масштабирования исходных срезов с использованием диагональной матрицы. Срезы данных — это столбцы матрицы F. Первые N-1 столбцов матрицы F обозначаются как X, а последние N-1 — как Y.

jobz

[in] Значение из перечисления ENUM_DMDQ_EIGV, которое определяет необходимость вычисления собственных векторов (мод Купмана).

jobr

[in] Значение из перечисления ENUM_DMD_RESIDUALS, которое определяет, будут ли вычисляться остатки.

jobq

[in] Значение из перечисления ENUM_DMDQ_Q, которое определяет, нужно ли явно вычислять и возвращать унитарную матрицу из QR-разложения матрицы срезов данных.

jobt

[in] Значение из перечисления ENUM_DMDQ_R, которое определяет нужно ли возвращать верхнетреугольную матрицу из QR-разложения матрицы срезов данных.

jobf

[in] Значение из перечисления ENUM_DMD_REFINE, которое определяет, следует ли сохранять информацию, необходимую для постобработки (например, вычисления уточненных векторов Ритца).

whtsvd

[in] Значение из перечисления ENUM_SVD_ALG, которое позволяет выбрать алгоритм SVD из библиотеки LAPACK.

nrnk

[in] Значение определяет способ вычисления числового ранга, т.е. способ усечения малых сингулярных значений входной матрицы X. При

NRNK = -1 :: i-е сингулярное значение sigma(i) усекается, если sigma(i) <= TOL*sigma(1). Рекомендуемый вариант.

При NRNK = -2 :: i-е сингулярное значение sigma(i) усекается, если sigma(i) <= TOL*sigma(i-1). Эта опция включена для исследований и разработок. Требует высокой точности SVD, что не всегда достигаемо.

© 2000-2025, MetaQuotes Ltd.

Числовой ранг может быть задан с помощью положительного значения NRNK следующим образом: 0 < NRNK <= N-1 :: будут использоваться не более NRNK наибольших сингулярных значений. Если число вычисленных ненулевых сингулярных значений меньше NRNK, то будут использоваться только эти ненулевые значения, а фактически используемая размерность будет меньше NRNK.

tol

[in] Допустимая погрешность для усечения малых сингулярных значений. 0 <= TOL < 1

eigen_values

[out] Вектор собственных значений размера K. Ведущие K (K<N) элементов в EIGS содержат вычисленные собственные значения (значения Ритца).

kb.

[out] Ортогональная матрица (фактор Q) из QR-разложения исходной матрицы срезов F.

R

[out] Верхнетреугольная матрица размером N на N, фактор R из QR-разложения матрицы F.

left_vectors

[out] Матрица левых сингулярных векторов. Первые K столбцов матрицы X содержат K главных левых сингулярных векторов представлений первых N-1 срезов в ортонормированном базисе, полученном из QR-разложения F. Чтобы поднять их в пространство левых сингулярных векторов U(:,1:K) в исходном пространстве данных, нужно предварительно умножить их на фактор Q из исходного QR-разложения.

Z

[out] Матрица размером M на K

Если JOBZ =='V', то Z содержит векторы Ритца. Z(:,i) — собственный вектор i-го значения Ритца; ||Z(:,i)||_2=1.

Если JOBZ == 'F', тогда Z(:,i) задаются неявно как Z*V, где Z содержит ортонормированную матрицу (произведение фактора Q из исходного QR-разложения и SVD/POD_basis, возвращаемого функцией DynamicModeDecomposition в X), а второй фактор (собственные векторы отношения Рэлея) находится в матрице V, возвращаемой функцией DynamicModeDecomposition. Таким образом, X(:,1:K)*V(:,i) — это собственный вектор, соответствующий EIGS(i). Столбцы V(1:K,1:K) являются вычисленными собственными векторами отношения Рэлея размером K на K.

residuals

[out] Остатки для K вычисленных пар Ритца.

B

[out] Матрица размером N на K (K<N).

Если JOBF =='R', B(1:N,1:K) содержит A*U(:,1:K) и может использоваться для вычисления уточненных векторов.

Если JOBF == 'E', B(1:N,1;K) содержит A*U(:,1:K)*W(1:K,1:K), которые являются векторами из точного DMD (Exact), с точностью до масштабирования по обратным собственным значениям.

В обоих случаях содержимое B можно восстановить до исходного пространства домножением на фактор Q из исходного QR-разложения. Здесь A означает сжатие базового оператора.

© 2000-2025, MetaQuotes Ltd.

Если JOBF =='N', то матрица B не используется.

V

[out] Матрица размером K на K (K<N). Содержит собственные векторы матрицы отношения Рэлея. Векторы, возвращаемые в Z, являются произведением Q из исходного QR-разложения.

S

[out] Матрица размером K на K (K<N). Массив S(1:K,1:K) используется для матричного отношения Рэлея. Содержимое перезаписывается во время разложения собственных значений с помощью GEEV.

Возвращаемое значение

Возвращает true в случае успеха или false, если произошла ошибка.

Примечание

Количество строк (M) матрицы не должно быть меньше количества столбцов (N).

ENUM_DMD_SCALE

Перечисление, которое определяет, масштабируются ли исходные срезы данных с помощью диагональной матрицы.

Идентификатор	Описание
DMDSCALE_S	'S' — матрицы срезов данных X и Y умножаются на диагональную матрицу D так, что X*D имеет единичные ненулевые столбцы.
DMDSCALE_C	'C' — срезы масштабируются так же, как и в варианте 'S'. Если обнаружено, что i-й столбец матрицы X является нулевым вектором, а соответствующий i-й столбец матрицы Y не равен нулю, то i-й столбец Y устанавливается равным нулю.
DMDSCALE_Y	'Y' — матрицы срезов данных X и Y умножаются на диагональную матрицу D так, что Y*D имеет единичные ненулевые столбцы.
DMDSCALE_N	'N' — масштабирование данных не предусмотрено.

ENUM_DMDQ_EIGV

Перечисление, которое определяет, будут ли вычисляться собственные векторы (моды Купмана).

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
DMDQEIGV_V	'V' — собственные векторы (моды Купмана) будут вычисляться.
DMDQEIGV_F	'F' — будут возвращены собственные векторы (моды Купмана) в факторизованной форме как произведение Z*V
DMDQEIGV_Q	'Q' — будут возвращены собственные векторы (моды Купмана) в факторизованной форме как произведение Q*Z
DMDQEIGV_N	'N' — собственные векторы не вычисляются

ENUM_DMD_RESIDUALS

Перечисление, которое определяет необходимость вычисления остатков.

Идентификатор	Описание
DMDRESIDUALS_R	'R' — вычислять остатки для собственных пар.
DMDRESIDUALS_N	'N' — не вычислять остатки.

ENUM_DMDQ_Q

Перечисление, которое указывает, следует ли явно вычислять и возвращать унитарную матрицу из QR-разложения.

Идентификатор	Описание
DMDQQ_Q	'Q' — вычисляется матрица Q из QR-разложения исходной матрицы срезов
DMDQQ_N	'N' — матрица Q явно не вычисляется

ENUM_DMDQ_R

Перечисление, которое указывает, следует ли возвращать верхнетреугольную матрицу из QR-разложения.

Идентификатор	Описание
DMDQR_R	'R' — вычисляется матрица R из QR-разложения исходной матрицы срезов
DMDQR_N	'N' — матрица R явно не вычисляется

© 2000-2025, MetaQuotes Ltd.

ENUM_DMD_REFINE

Перечисление, которое определяет, следует ли хранить информацию, необходимую для постобработки.

Идентификатор	Описание
DMDREFINE_R	'R' — матрица, необходимая для уточнения векторов Ритца, вычисляется и сохраняется в массиве B.
DMDREFINE_E	'E' — немасштабированные собственные векторы точного DMD вычисляются и возвращаются в массиве B.
DMDREFINE_N	'N' — данные для уточнения собственных векторов не вычисляются.

ENUM_SVD_ALG

Перечисление для выбора алгоритма SVD.

Идентификатор	Описание
SVDALG_1	1: GESVD (алгоритм QR SVD)
SVDALG_2	2: GESDD (алгоритм Divide and Conquer)
SVDALG_3	3: GESVDQ (предобусловленный алгоритм QR SVD; этот и следующий варианты самые точные)
SVDALG_4	4: GEJSV (предобусловленный алгоритм Jacobi SVD; этот и предыдущий варианты самые точные)

© 2000-2025, MetaQuotes Ltd.

Singular Spectrum Analysis

Раздел содержит функции для разложения матрицы на три компонента: ортогональные матрицы и диагональную матрицу сингулярных значений. SVD применяется для решения различных задач линейной алгебры, таких как уменьшение размерности данных, сжатие изображений, решение систем уравнений, а также анализ данных и оптимизация. Основные функции позволяют вычислять сингулярные значения и векторы, восстанавливать матрицы и производить аппроксимацию матриц с уменьшенной ранговой точностью.

Функция	Действие
SingularSpectrumAnalysisSpectrum	Функция-метод расчёта относительных вкладов спектральных компонент по собственным значениям.
SingularSpectrumAnalysisForecast	Функция-метод расчёта реконструированных и прогнозируемых данных с использованием спектральных компонент входного временного ряда.
SingularSpectrumAnalysisReconstructComponents	Функция-метод расчёта реконструированных компоненент входного временного ряда и их вклады.
SingularSpectrumAnalysisReconstructSeries	Функция-метод расчёта реконструированного временного ряда с использованиеми первых component_count компонент.

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisSpectrum

Функция-метод расчёта относительных вкладов спектральных компонент по собственным значениям.

Вычисления для типа vector<double>

```
bool  vector::SingularSpectrumAnalysisSpectrum(
   ulong   window_length,      // размер окна для построения траекторной матрицы
   vector& spectrum            // вектор вкладов компонент во входной ряд (спектр SSA)
   );
```

Вычисления для типа vector<complex>

```
bool  vector::SingularSpectrumAnalysisSpectrum(
   ulong   window_length,      // размер окна для построения траекторной матрицы
   vectorc& spectrum           // вектор вкладов компонент во входной ряд (спектр SSA)
   );
```

Параметры

window_length

[in] Размер окна для построения траекторной матрицы, количество компонент, на которые должен быть разложен входной временной ряд.

spectrum

[out] Вектор вкладов компонент во входной ряд — собственные значения ковариационной матрицы входного временного ряда.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Значение параметра *window_length* должно быть меньше размера входного временного ряда. Для построения полноценной траекторной матрицы считается оптимальным размер примерно равный половине размера входного временного ряда.

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisForecast

Функция-метод расчёта реконструированных и прогнозируемых данных с использованием спектральных компонент входного временного ряда.

Вычисления для типа vector<double>

```
bool  vector::SingularSpectrumAnalysisForecast(
   ulong    window_length,          // размер окна для построения траекторной матрицы
   ulong    component_count,        // количество компонент, используемых для прогноза
   ulong    forecast_horizon,       // количество точек, которые нужно спрогнозировать
   vector&  forecast                // вектор, состоящий из реконструированных и спрогн
   );
```

Вычисления для типа vector<complex>

```
bool  vector::SingularSpectrumAnalysisForecast(
   ulong    window_length,          // размер окна для построения траекторной матрицы
   ulong    component_count,        // количество компонент, используемых для прогноза
   ulong    forecast_horizon,       // количество точек, которые нужно спрогнозировать
   vectorc& forecast                // вектор, состоящий из реконструированных и спрогн
   );
```

Параметры

window_length

[in] Размер окна для построения траекторной матрицы, количество компонент, на которые должен быть разложен входной временной ряд.

component_count

[in] Количество компонент, используемых для прогноза.

forecast_horizon

[in] Количество точек, которые нужно спрогнозировать.

forecast

[out] Объединение реконструированных по *component_count* точек плюс *forecast_horizon* точек, спрогнозированных с помощью первых *component_count* компонент. Таким образом вектор *forecast* имеет размер (T+forecast_horizon), где T — это длина входного ряда.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Значение параметра *window_length* должно быть меньше размера входного временного ряда. Для построения полноценной траекторной матрицы считается оптимальным размер примерно равный половине размера входного временного ряда.

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisReconstructComponents

Функция-метод расчёта реконструированных компонент входного временного ряда и их вклады.

Вычисления для типа vector<double> и matrix<double>

```
bool  vector::SingularSpectrumAnalysisReconstructComponents(
   ulong    window_length,      // размер окна для построения траекторной матрицы
   matrix&  components,         // матрица реконструированных компонент
   vector&  contributions       // вектор вкладов компонент во входной ряд
   );
```

Вычисления для типа vector<complex> и matrix<complex>

```
bool  vector::SingularSpectrumAnalysisReconstructComponents(
   ulong    window_length,      // размер окна для построения траекторной матрицы
   matrixc& components,         // матрица реконструированных компонент
   vectorc& contributions       // вектор вкладов компонент во входной ряд
   );
```

Параметры

window_length

[in] Размер окна для построения траекторной матрицы, количество компонент, на которые должен быть разложен входной временной ряд.

components

[out] Матрица реконструированных компонент, где каждая колонка описывает отдельную компоненту.

contributions

[out] Вектор вкладов компонент во входной ряд (собственные значения ковариационной матрицы входного временного ряда).

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Значение параметра *window_length* должно быть меньше размера входного временного ряда. Для построения полноценной траекторной матрицы считается оптимальным размер примерно равный половине размера входного временного ряда.

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisReconstructSeries

Функция-метод расчёта реконструированного временного ряда с использованием первых *component_count* компонент.

Вычисления для типа vector<double>

```
bool  vector::SingularSpectrumAnalysisReconstructSeries(
   ulong    window_length,       // размер окна для построения траекторной матрицы
   ulong    component_count,     // количество компонент, используемых для реконструкц
   vector&  reconstructed        // реконструированный временной ряд
   );
```

Вычисления для типа vector<complex>

```
bool  vector::SingularSpectrumAnalysisReconstructSeries(
   ulong    window_length,       // размер окна для построения траекторной матрицы
   ulong    component_count,     // количество компонент, используемых для реконструкц
   vectorc& reconstructed        // реконструированный временной ряд
   );
```

Параметры

window_length

[in] Размер окна для построения траекторной матрицы, количество компонент, на которые должен быть разложен входной временной ряд.

component_count

[out] Количество компонент, используемых для реконструкции временного ряда.

reconstructed

[out] Вектор, содержащий реконструированный выходной ряд.

Возвращаемое значение

Возвращает true в случае успеха или false в случае ошибки.

Примечание

Значение параметра *window_length* должно быть меньше размера входного временного ряда. Для построения полноценной траекторной матрицы считается оптимальным размер примерно равный половине размера входного временного ряда.

© 2000-2025, MetaQuotes Ltd.

Reductions

Функции, предназначенные для приведения матриц к специальным формам с использованием ортогональных преобразований. Эти методы широко применяются в численных вычислениях, например, в спектральном анализе, решении задач на собственные значения и сингулярном разложении матриц.

Методы позволяют эффективно преобразовывать матрицы, что необходимо для решения задач линейной алгебры, включая вычисление сингулярного разложения (SVD) и нахождение собственных значений симметричных матриц. Использование LAPACK-функций гарантирует высокую производительность и точность вычислений.

Функция	Действие
ReduceToBidiagonal	Преобразует произвольную вещественную или комплексную матрицу A размера m×n к верхнему или нижнему бидиагональному виду B с помощью ортогонального преобразования: Q**T * A * P = B. При m >= n матрица B будет верхней бидиагональной; при m < n — нижней. Функция Lapack GEBRD.
ReflectBidiagonalToQP	Генерирует ортогональные матрицы Q и P**T (или P**H для комплексных типов), определяемые методом ReduceToBidiagonal при приведении вещественной или комплексной матрицы A к бидиагональному виду: A = Q * B * P**T. Матрицы Q и P**T представляют собой произведения элементарных отражений H(i) или G(i) соответственно. Функции Lapack ORGBR и UNGBR.
ReduceSymmetricToTridiagonal	Преобразует вещественную симметричную или комплексную эрмитову матрицу A в трехдиагональная форму B с помощью ортогонального преобразования подобия: Q**T * A * Q = B. Lapack-функции SYTRD и HETRD.
ReflectTridiagonalToQ	Генерирует ортогональную матрицу Q, которая представляется в виде произведения n-1 элементарных отражений порядка n, как при использовании метода ReduceSymmetricToTridiagonal

© 2000-2025, MetaQuotes Ltd.

ReduceToBidiagonal

Преобразует произвольную вещественную или комплексную матрицу A размера m×n к верхнему или нижнему бидиагональному виду B с помощью ортогонального преобразования: Q**T * A * P = B. При m >= n матрица B будет верхней бидиагональной; при m < n — нижней. Функция Lapack GEBRD.

Вычисления для типа matrix<double>

```
bool  matrix::ReduceToBidiagonal(
   matrix&          B,             // бидиагональная матрица
   matrix&          reflect_qp,    // q- и p-отражатели
   vector&          tau_q,         // скалярные коэффициенты элементарных отражателей Q
   vector&          tau_p          // скалярные коэффициенты элементарных отражателей P
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::ReduceToBidiagonal(
   matrixf&         B,             // бидиагональная матрица
   matrixf&         reflect_qp,    // q- и p-отражатели
   vectorf&         tau_q,         // скалярные коэффициенты элементарных отражателей Q
   vectorf&         tau_p          // скалярные коэффициенты элементарных отражателей P
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::ReduceToBidiagonal(
   matrixc&         B,             // бидиагональная матрица
   matrixc&         reflect_qp,    // q- и p-отражатели
   vectorc&         tau_q,         // скалярные коэффициенты элементарных отражателей Q
   vectorc&         tau_p          // скалярные коэффициенты элементарных отражателей P
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::ReduceToBidiagonal(
   matrixcf&        B,             // бидиагональная матрица
   matrixcf&        reflect_qp,    // q- и p-отражатели
   vectorcf&        tau_q,         // скалярные коэффициенты элементарных отражателей Q
   vectorcf&        tau_p          // скалярные коэффициенты элементарных отражателей P
   );
```

Параметры

B

[out] Верхняя или нижняя бидиагональная матрица.

reflect_qp

[out] Преобразованная матрица A. При m >= n диагональ и первая наддиагональ перезаписываются верхней бидиагональной матрицей B; элементы ниже диагонали вместе с вектором tau_q представляют ортогональную матрицу Q как произведение элементарных отражателей, а элементы выше первой наддиагонали вместе с вектором tau_p представляют

© 2000-2025, MetaQuotes Ltd.

ортогональную матрицу P как произведение элементарных отражателей. При m < n диагональ и первая поддиагональ перезаписываются нижней бидиагональной матрицей B; при этом элементы ниже первой поддиагонали вместе с вектором tau_q представляют ортогональную матрицу Q как произведение элементарных отражателей, а элементы выше диагонали с вектором tau_p представляют ортогональную матрицу P как произведение элементарных отражателей.

tau_q

[out] Вектор скалярных множителей элементарных отражателей, представляющих ортогональную матрицу Q.

tau_p

[out] Вектор скалярных множителей элементарных отражателей, представляющих ортогональную матрицу P.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Матрицы Q и P представляются в виде произведений элементарных рефлекторов:

При m >= n

Q = H(1) H(2) . . . H(n) и P = G(1) G(2) . . . G(n-1)

Каждый H(i) и G(i) имеет вид:

H(i) = I - tauq * v * v**T и G(i) = I - taup * u * u**T

где tauq и taup — это скаляры, а v и u — векторы;

v(1:i-1) = 0, v(i) = 1, а v(i+1:m) сохраняется на выходе в матрице A(i+1:m,i);

u(1:i) = 0, u(i+1) = 1, а u(i+2:n) сохраняется на выходе в матрице A(i,i+2:n);

tauq сохраняется в массиве *tau_q*(i), а taup — в *tau_p*(i).

При m < n

Q = H(1) H(2) . . . H(m-1) и P = G(1) G(2) . . . G(m)

Каждый H(i) и G(i) имеет вид:

H(i) = I - tauq * v * v**T и G(i) = I - taup * u * u**T

где tauq и taup — это скаляры, а v и u — векторы;

v(1:i) = 0, v(i+1) = 1, а v(i+2:m) сохраняется на выходе в матрице A(i+2:m,i);

u(1:i-1) = 0, u(i) = 1, а u(i+1:n) сохраняется на выходе в матрице A(i,i+1:n);

tauq сохраняется в массиве *tau_q*(i), а taup — в *tau_p*(i).

© 2000-2025, MetaQuotes Ltd.

Содержимое матрицы A при выходе (выходная матрица *reflect_qp*) можно проиллюстрировать следующими примерами:

```
m = 6 and n = 5 (m > n):          m = 5 and n = 6 (m < n):
   (  d   e   u1  u1  u1 )           (  d   u1  u1  u1  u1  u1 )
   (  v1  d   e   u2  u2 )           (  e   d   u2  u2  u2  u2 )
   (  v1  v2  d   e   u3 )           (  v1  e   d   u3  u3  u3 )
   (  v1  v2  v3  d   e  )           (  v1  v2  e   d   u4  u4 )
   (  v1  v2  v3  v4  d  )           (  v1  v2  v3  e   d   u5 )
   (  v1  v2  v3  v4  v5 )
```

где d и e обозначают диагональные и недиагональные элементы матрицы B, vi — элемент вектора, определяющего H(i), а ui — элемент вектора, определяющего G(i).

Матрицы Q и P могут быть получены с помощью метода ReflectBidiagonalToQP.

© 2000-2025, MetaQuotes Ltd.

ReflectBidiagonalToQP

Генерирует ортогональные матрицы Q и P**T (или P**H для комплексных типов), определяемые методом ReduceToBidiagonal при приведении вещественной или комплексной матрицы A к бидиагональному виду: A = Q * B * P**T. Матрицы Q и P**T представляют собой произведения элементарных отражений H(i) или G(i) соответственно. Функции Lapack ORGBR и UNGBR.

Входными данными является преобразованная матрица reflect_qp того же размера m×n, что и исходная матрица A.

Вычисления для типа matrix<double>

```
bool  matrix::ReflectBidiagonalToQP(
   vector&          tau_q,         // скалярные коэффициенты элементарных отражателей Q
   vector&          tau_p,         // скалярные коэффициенты элементарных отражателей P
   matrix&          Q,             // матрица Q
   matrix&          PT             // транспонированная матрица P
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::ReflectBidiagonalToQP(
   vectorf&         tau_q,         // скалярные коэффициенты элементарных отражателей Q
   vectorf&         tau_p,         // скалярные коэффициенты элементарных отражателей P
   matrixf&         Q,             // матрица Q
   matrixf&         PT             // транспонированная матрица P
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::ReflectBidiagonalToQP(
   vectorc&         tau_q,         // скалярные коэффициенты элементарных отражателей Q
   vectorc&         tau_p,         // скалярные коэффициенты элементарных отражателей P
   matrixc&         Q,             // матрица Q
   matrixc&         PH             // эрмитово сопряженная матрица P
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::ReflectBidiagonalToQP(
   vectorcf&        tau_q,         // скалярные коэффициенты элементарных отражателей Q
   vectorcf&        tau_p,         // скалярные коэффициенты элементарных отражателей P
   matrixcf&        Q,             // матрица Q
   matrixcf&        PH             // эрмитово сопряженная матрица P
   );
```

Параметры

tau_q

[in] Вектор скалярных множителей элементарных отражателей, представляющих ортогональную матрицу Q.

tau_p

© 2000-2025, MetaQuotes Ltd.

[in] Вектор скалярных множителей элементарных отражателей, представляющих ортогональную матрицу P.

Q

[out] Ортогональная матрица Q.

PT (PH)

[out] Транспонированная (или эрмитово сопряженная) матрица P.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

Если m >= n, то матрица Q имеет размер m×n, а матрица PT имеет размер имеет размер n×n.

Если m < n, то матрица Q имеет размер m×m, а матрица PT имеет размер имеет размер m×n.

© 2000-2025, MetaQuotes Ltd.

ReduceSymmetricToTridiagonal

Преобразует вещественную симметричную или комплексную эрмитову матрицу A в трехдиагональную форму B с помощью ортогонального преобразования подобия: Q**T * A * Q = B. Lapack-функции SYTRD и HETRD.

Вычисления для типа matrix<double>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrix&          B,             // трехдиагональная матрица
   matrix&          reflect_q,     // q-отражатели
   vector&          tau_q          // скалярные коэффициенты элементарных отражателей Q
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrixf&         B,             // трехдиагональная матрица
   matrixf&         reflect_q,     // q-отражатели
   vectorf&         tau_q          // скалярные коэффициенты элементарных отражателей Q
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrixc&         B,             // трехдиагональная матрица
   matrixc&         reflect_q,     // q-отражатели
   vectorc&         tau_q          // скалярные коэффициенты элементарных отражателей Q
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrixcf&        B,             // трехдиагональная матрица
   matrixcf&        reflect_q,     // q- и p-отражатели
   vectorcf&        tau_q          // скалярные коэффициенты элементарных отражателей Q
   );
```

Параметры

B

[out] Симметричная (или эрмитова) трехдиагональная матрица.

reflect_q

[out] Верхняя или нижняя треугольная матрица в зависимости от входной матрицы A. Если верхняя, диагональ и первая наддиагональ матрицы A перезаписываются соответствующими элементами трехдиагональной матрицы B, а элементы выше первой наддиагонали вместе с массивом *tau_q* представляют ортогональную матрицу Q в виде произведения элементарных отражателей. Если нижняя, диагональ и первая поддиагональ матрицы A перезаписываются соответствующими элементами трехдиагональной матрицы B, а элементы ниже первой поддиагонали вместе с массивом *tau_q* представляют ортогональную матрицу Q в виде произведения элементарных отражателей.

© 2000-2025, MetaQuotes Ltd.

tau_q

[out] Вектор скалярных множителей элементарных отражателей, представляющих ортогональную матрицу Q.

Возвращаемое значение

Возвращает true в случае успеха, иначе false в случае ошибки.

Примечание

В верхнем случае матрица Q представляется как произведение элементарных отражателей

Q = H(n-1) . . . H(2) H(1)

Каждый H(i) имеет вид

H(i) = I - tau * v * v**T

где tau — это скаляр, а v — это вектор, при этом v(i+1:n) = 0 и v(i) = 1; v(1:i-1) сохраняется в матрице A(1:i-1,i+1), а tau — в массиве *tau_q*(i).

В нижнем случае матрица Q представляется как произведение элементарных отражателей

Q = H(1) H(2) . . . H(n-1)

Каждый H(i) имеет вид

H(i) = I - tau * v * v**T

где tau — это скаляр, а v — это вектор, при этом v(1:i) = 0 и v(i+1) = 1; v(i+2:n) сохраняется в матрице A(i+2:n,i), а tau — в массиве *tau_q*(i). */

Содержимое матрицы A при выходе (выходная матрица *reflect_q*) можно проиллюстрировать следующими примерами при n = 5:

```
      upper case:                          lower case:

      (  d   e   v2  v3  v4 )              (  d                  )
      (      d   e   v3  v4 )              (  e   d              )
      (          d   e   v4 )              (  v1  e   d          )
      (              d   e  )              (  v1  v2  e   d      )
      (                  d  )              (  v1  v2  v3  e   d  )
```

где d и e обозначают диагональные и недиагональные элементы матрицы B, а vi — элемент вектора, определяющего H(i).

Матрица Q может быть получена с помощью метода ReflectTridiagonalToQ.

© 2000-2025, MetaQuotes Ltd.

ReflectTridiagonalToQ

Генерирует ортогональную матрицу Q, которая представляется в виде произведения n-1 элементарных отражений порядка n, как при использовании метода ReduceSymmetricToTridiagona:

Если верхняя, Q = H(n-1) . . . H(2) H(1),

если нижняя, Q = H(1) H(2) . . . H(n-1).

Lapack-функции ORGTR, UNGTR.

Входными данными является преобразованная матрица reflect_q того же размера n×n, что и исходная матрица A.

Вычисления для типа matrix<double>

```
bool  matrix::ReflectTridiagonalToQ(
   vector&         tau_q,        // скалярные коэффициенты элементарных отражателей Q
   matrix&         Q             // матрица Q
   );
```

Вычисления для типа matrix<float>

```
bool  matrix::ReflectTridiagonalToQ(
   vectorf&        tau_q,        // скалярные коэффициенты элементарных отражателей Q
   matrixf&        Q             // матрица Q
   );
```

Вычисления для типа matrix<complex>

```
bool  matrix::ReflectTridiagonalToQ(
   vectorc&        tau_q,        // скалярные коэффициенты элементарных отражателей Q
   matrixc&        Q             // матрица Q
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrix::ReflectTridiagonalToQ(
   vectorcf&       tau_q,        // скалярные коэффициенты элементарных отражателей Q
   matrixcf&       Q             // матрица Q
   );
```

Параметры

tau_q

[in] Вектор скалярных множителей элементарных отражателей, представляющих ортогональную матрицу Q.

Q

[out] Ортогональная матрица Q.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успеха, иначе false в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Преобразование данных

Группа функций, обеспечивающих преобразование данных из одного формата в данные другого формата.

Особо следует отметить функцию NormalizeDouble(), которая обеспечивает требуемую точность представления цены. В торговых операциях нельзя использовать ненормализованные цены, чья точность превышает требуемую торговым сервером хотя бы на один знак.

Функция	Действие
CharToString	Преобразование кода символа в односимвольную строку
DoubleToString	Преобразование числового значения в текстовую строку с указанной точностью
EnumToString	Преобразование значения перечисления любого типа в строку
NormalizeDouble	Округление числа с плавающей точкой до указанной точности
StringToDouble	Преобразование строки, содержащей символьное представление числа, в число типа double
StringToInteger	Преобразование строки, содержащей символьное представление числа, в число типа long
StringToTime	Преобразование строки, содержащей время и/или дату в формате "yyyy.mm.dd [hh:mi]", в число типа datetime
TimeToString	Преобразование значения, содержащего время в секундах, прошедшее с 01.01.1970, в строку формата "yyyy.mm.dd hh:mi"
IntegerToString	Преобразование значения целого типа в строку указанной длины
ShortToString	Преобразование код символа (unicode) в односимвольную строку
ShortArrayToString	Копирует часть массива в строку
StringToShortArray	Посимвольно копирует строку в указанное место массива типа ushort
CharArrayToString	Преобразует код символа (ansi) в односимвольную строку
StringToCharArray	Посимвольно копирует преобразованную из Unicode в ANSI строку в указанное место массива типа uchar
CharArrayToStruct	Копирует POD-структуру в массив типа uchar
StructToCharArray	Копирует POD-структуру в массив типа uchar
ColorToARGB	Преобразует тип color в тип uint для получения ARGB-представления цвета.
ColorToString	Преобразует значение цвета в строку вида "R,G,B"
StringToColor	Преобразует строку типа "R,G,B" или строку, содержащую наименование цвета, в значение типа color

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
StringFormat	Преобразует число в строку в соответствие с заданным форматом

Смотри также

Использование кодовой страницы

© 2000-2025, MetaQuotes Ltd.

CharToString

Преобразование кода символа в односимвольную строку.

```
string  CharToString(
   uchar  char_code      // числовой код символа
   );
```

Параметры

char_code

[in] Код символа ANSI.

Возвращаемое значение

Строка, содержащая символ ANSI.

Пример:

```
string ExtStrArray[224];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- в цикле от кода символа 32(пробел) до кода 255(UCHAR_MAX)
//--- заполним массив кодами символов, преобразованными в строку в соответствии с теку
   for(int i=32; i<=UCHAR_MAX; i++)
      ExtStrArray[i-32]=CharToString((uchar)i);

//--- распечатаем в журнале заголовок и таблицу символов
   Print("Table of symbols:");
   ArrayPrint(ExtStrArray,_Digits," | ");
   /*
   результат:
   Table of symbols:
   [  0] " " | "!" | """ | "#" | "$" | "%" | "&" | "'" | "(" | ")" | "*" | "+" | ","
   [ 25] "9" | ":" | ";" | "<" | "=" | ">" | "?" | "@" | "A" | "B" | "C" | "D" | "E"
   [ 50] "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "[" | "\" | "]" | "^"
   [ 75] "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w"
   [100] "„" | "…" | "†" | "‡" | "€" | "‰" | "Љ" | "‹" | "Њ" | "Ќ" | "Ћ" | "Џ" | "Ђ"
   [125] "ќ" | "ћ" | "џ" | " " | "ў" | "Ў" | "Ј" | "¤" | "Ґ" | "¦" | "§" | "Ё" | "©"
   [150] "Ґ" | "·" | "ё" | "№" | "є" | "»" | "ј" | "S" | "s" | "і" | "А" | "Б" | "В"
   [175] "П" | "Р" | "С" | "Т" | "У" | "Ф" | "Х" | "Ц" | "Ч" | "Ш" | "Щ" | "Ъ" | "Ы"
   [200] "и" | "й" | "к" | "л" | "м" | "н" | "о" | "п" | "р" | "с" | "т" | "у" | "ф"
   */
  }
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

StringToCharArray, ShortToString, StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

CharArrayToString

Копирует и преобразует часть массива типа uchar в возвращаемую строку.

```
string  CharArrayToString(
   uchar    array[],             // массив
   int      start=0,             // начальная позиция в массиве
   int      count=-1             // количество символов
   uint     codepage=CP_ACP      // кодовая страница
   );
```

Параметры

array[]

[in] Массив типа uchar.

start=0

[in] Позиция, с которой начинается копирование. По умолчанию 0.

count=-1

[in] Количество элементов массива для копирования. Определяет длину результатной строки. По умолчанию -1, что означает копирование до конца массива, либо до встречи терминального 0.

codepage=CP_ACP

[in] Значение кодовой страницы. Для наиболее употребимых кодовых страниц предусмотрены соответствующие константы.

Возвращаемое значение

Строка.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создаём массив с кодами символов
   uchar array[]= { 84, 104, 105, 115,  32, 105, 115,  32,  97,  32,
                    116, 101, 115, 116,  32, 111, 102,  32, 116, 104,
                    101,  32,  67, 104,  97, 114,  65, 114, 114,  97,
                    121,  84, 111,  83, 116, 114, 105, 110, 103,  40,
                     41,  32, 102, 117, 110,  99, 116, 105, 111, 110
                   };
//--- распечатываем в журнале массив кодов, преобразованных в строку
   Print(CharArrayToString(array));

//--- создаём массив с кодами символов и терминальным нулём
```

© 2000-2025, MetaQuotes Ltd.

```
   uchar array_z[]= { 84, 104, 105, 115,  32, 105, 115,  32,  97,  32,
                     116, 101, 115, 116,   0,  32, 111, 102,  32, 116,
                     104, 101,  32,  67, 104,  97, 114,  65, 114, 114,
                      97, 121,  84, 111,  83, 116, 114, 105, 110, 103,
                      40,  41,  32, 102, 117, 110,  99, 116, 105, 111, 110
                    };
//--- распечатываем в журнале второй массив кодов с терминальным нулём
   PrintFormat("%s ...", CharArrayToString(array_z));

   /*
   результат:
   This is a test of the CharArrayToString() function
   This is a test ...
   */
  }
```

Смотри также

StringToCharArray, ShortArrayToString, Использование кодовой страницы

© 2000-2025, MetaQuotes Ltd.

CharArrayToStruct

Копирует массив типа uchar в POD-структуру.

```
bool  CharArrayToStruct(
   void&          struct_object,     // структура
   const uchar&   char_array[],      // массив
   uint           start_pos=0        // начальная позиция в массиве
   );
```

Параметры

struct_object

[in] Ссылка на любой тип POD-структуры (структуры, содержащей только простые типы данных).

char_array[]

[in] Массив типа uchar.

start_pos=0

[in] Позиция в массиве, начиная с которой будут скопированы данные.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
#define DATA_TOTAL 4

MqlRates ExtDataRates[DATA_TOTAL];
uchar    ExtCharArray[];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- копируем в uchar-массив данные четырёх последних баров
   ResetLastError();
   for(int i=0; i<DATA_TOTAL; i++)
     {
      if(!CopyRatesToCharArray(i, ExtCharArray))
         return;
     }

//--- копируем в цикле по количеству данных из uchar-массива все имеющиеся данные в ма
   for(int i=0; i<DATA_TOTAL; i++)
     {
      ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
    if(!CharArrayToStruct(ExtDataRates[i], ExtCharArray, sizeof(MqlRates)*i))
      {
       Print("CharArrayToStruct() failed. Error code: ", GetLastError());
       return;
      }
    //--- при успешном копировании данных из uchar-массива в структуру MqlRates
    //--- распечатаем в журнал данные, скопированные в структуру
    MqlRatesPrint(ExtDataRates[i]);
   }

/*
результат:
MqlRates data:
 Time = 2024.02.21 09:00;
 Open = 1.08116;
 High = 1.08141;
 Low = 1.08062;
 Close = 1.08124;
 Tick volume = 5344;
 Spread = 1;
 Real volume = 0

MqlRates data:
 Time = 2024.02.21 08:00;
 Open = 1.08149;
 High = 1.08153;
 Low = 1.08073;
 Close = 1.08114;
 Tick volume = 3607;
 Spread = 2;
 Real volume = 0

MqlRates data:
 Time = 2024.02.21 07:00;
 Open = 1.08143;
 High = 1.08165;
 Low = 1.08122;
 Close = 1.08150;
 Tick volume = 2083;
 Spread = 0;
 Real volume = 0

MqlRates data:
 Time = 2024.02.21 06:00;
 Open = 1.08178;
 High = 1.08190;
 Low = 1.08132;
 Close = 1.08142;
 Tick volume = 1733;
```

© 2000-2025, MetaQuotes Ltd.

```
    Spread = 0;
    Real volume = 0
   */
  }
//+------------------------------------------------------------------+
//| Копирует по указанному индексу данные бара                       |
//| через структуру MqlRates в uchar-массив                          |
//+------------------------------------------------------------------+
bool CopyRatesToCharArray(const int index, uchar &data_array[])
  {
   MqlRates rates[1];
   uint     start=sizeof(MqlRates);

   ResetLastError();
   if(CopyRates(Symbol(), PERIOD_CURRENT, index, 1, rates)!=1)
     {
      Print("CopyRates() failed. Error code: ", GetLastError());
      return(false);
     }
   if(!StructToCharArray(rates[0], data_array, start*index))
     {
      Print("StructToCharArray() failed. Error code: ", GetLastError());
      return(false);
     }

   return(true);
  }
//+------------------------------------------------------------------+
//| Распечатывает в журнал поля структуры MqlRates                   |
//+------------------------------------------------------------------+
void MqlRatesPrint(MqlRates &rates)
  {
//--- создаём строку вывода
   string text=StringFormat(" Time = %s;\n"
                            " Open = %.*f;\n"
                            " High = %.*f;\n"
                            " Low = %.*f;\n"
                            " Close = %.*f;\n"
                            " Tick volume = %I64u;\n"
                            " Spread = %d;\n"
                            " Real volume = %I64u",
                            TimeToString(rates.time),
                            _Digits, rates.open,
                            _Digits, rates.high,
                            _Digits, rates.low,
                            _Digits, rates.close,
                            rates.tick_volume,
                            rates.spread,
                            rates.real_volume);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- распечатываем в журнале заголовок и строку вывода
   Print("MqlRates data:\n", text, "\n");
  }
```

Смотри также

StringToCharArray, ShortArrayToString, StructToCharArray, Использование кодовой страницы, FileReadStruct, Объединение (union), MathSwap

© 2000-2025, MetaQuotes Ltd.

StructToCharArray

Копирует POD-структуру в массив типа uchar.

```
bool  StructToCharArray(
   const void&  struct_object,      // структура
   uchar&       char_array[],       // массив
   uint         start_pos=0         // начальная позиция в массиве
   );
```

Параметры

struct_object

[in] Ссылка на любой тип POD-структуры (структуры, содержащей только простые типы данных).

char_array[]

[in] Массив типа uchar.

start_pos=0

[in] Позиция в массиве, начиная с которой будут добавлены скопированные данные.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

При копировании динамический массив автоматически расширяется (ArrayResize), если в нем недостаточно места. Если расширить массива до необходимой величины не удалось, функция вернет ошибку.

Пример:

```
uchar    ExtCharArray[];
MqlRates ExtRates[1];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- копируем в структуру MqlRates данные одного текущего бара
   if(CopyRates(Symbol(), PERIOD_CURRENT, 0, 1, ExtRates)!=1)
     {
      Print("CopyRates() failed. Error code: ", GetLastError());
      return;
     }

//--- распечатаем в журнале полученные в структуру MqlRates данные
   Print("Data in the structure immediately after receiving it:");
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlRatesPrint(ExtRates[0]);

//--- скопируем структуру в массив типа uchar
   ResetLastError();
   if(!StructToCharArray(ExtRates[0], ExtCharArray))
     {
      Print("StructToCharArray() failed. Error code: ", GetLastError());
      return;
     }

//--- очищаем структуру
   ZeroMemory(ExtRates[0]);
//--- распечатаем данные в структуре после очистки
   Print("\nData in the structure after ZeroMemory():");
   MqlRatesPrint(ExtRates[0]);

//--- теперь скопируем данные из uchar-массива в структуру MqlRates
   if(!CharArrayToStruct(ExtRates[0], ExtCharArray))
     {
      Print("CharArrayToStruct() failed. Error code: ", GetLastError());
      return;
     }
//--- распечатаем данные в структуре MqlRates после копирования их из uchar-массива
   Print("\nData in the MqlRates structure after copying it from a uchar array:");
   MqlRatesPrint(ExtRates[0]);

   /*
   результат:

   Data in the structure immediately after receiving it:
   MqlRates data:
    Time = 2024.02.21 07:00;
    Open = 1.08143;
    High = 1.08158;
    Low = 1.08122;
    Close = 1.08137;
    Tick volume = 1341;
    Spread = 4;
    Real volume = 0

   Data in the structure after ZeroMemory():
   MqlRates data:
    Time = 0;
    Open = 0.00000;
    High = 0.00000;
    Low = 0.00000;
    Close = 0.00000;
    Tick volume = 0;
    Spread = 0;
```

© 2000-2025, MetaQuotes Ltd.

```
    Real volume = 0

   Data in the MqlRates structure after copying it from a uchar array:
   MqlRates data:
    Time = 2024.02.21 07:00;
    Open = 1.08143;
    High = 1.08158;
    Low = 1.08122;
    Close = 1.08137;
    Tick volume = 1341;
    Spread = 4;
    Real volume = 0
   */
  }
//+------------------------------------------------------------------+
//| Распечатывает в журнал поля структуры MqlRates                   |
//+------------------------------------------------------------------+
void MqlRatesPrint(MqlRates &rates)
  {
//--- создаём строку вывода
   string text=StringFormat(" Time = %s;\n"
                            " Open = %.*f;\n"
                            " High = %.*f;\n"
                            " Low = %.*f;\n"
                            " Close = %.*f;\n"
                            " Tick volume = %I64u;\n"
                            " Spread = %d;\n"
                            " Real volume = %I64u",
                            TimeToString(rates.time),
                            _Digits, rates.open,
                            _Digits, rates.high,
                            _Digits, rates.low,
                            _Digits, rates.close,
                            rates.tick_volume,
                            rates.spread,
                            rates.real_volume);
//--- распечатываем в журнале заголовок и строку вывода
   Print("MqlRates data:\n", text);
  }
```

Смотри также

StringToCharArray, ShortArrayToString, CharArrayToStruct, Использование кодовой страницы, FileWriteStruct, Объединение (union), MathSwap

© 2000-2025, MetaQuotes Ltd.

ColorToARGB

Преобразует тип color в тип uint для получения ARGB-представления цвета. ARGB формат цвета используется при создании графического ресурса, вывода текста и в классе стандартной библиотеки CCanvas.

```
uint  ColorToARGB(
   color  clr,          // преобразуемый цвет в формате color
   uchar  alpha=255     // альфа-канал, управляющий прозрачностью цвета
   );
```

Параметры

clr

[in] Значение цвета в переменной типа color.

alpha

[in] Значение альфа-канала, для получения цвета в формате ARGB. Задается значением от 0 (цвет накладываемого пикселя совсем не меняет отображения нижележащего пикселя) до 255 (цвет накладывается полностью и перекрывает собою цвет нижележащего пикселя). Прозрачность цвета в процентом выражении вычисляется как (1-alpha/255)*100%, то есть чем меньше значение альфа-канала, тем более прозрачен цвет.

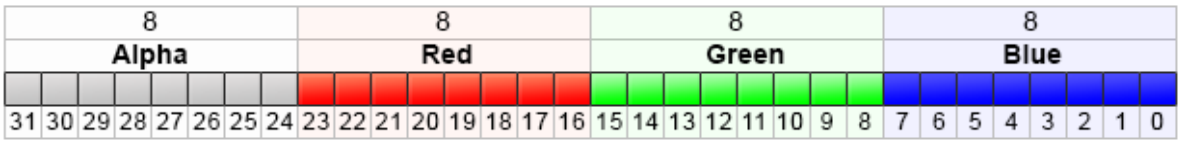
Возвращаемое значение

Представление цвета в формате ARGB, где в четырех байтах типа uint прописаны по порядку значения Alfa, Red, Green, Blue (альфа-канал, красный, зеленый, голубой).

Примечание

Основным общеупотребимым форматом для описания цвета пикселя на экране дисплея в компьютерной графике является RGB, где по названиям базовых цветов задаются красная (Red), зеленая (Green) и синяя (Blue) компоненты цвета. Каждая компонента описывается одним байтом, задающим насыщенность данного цвета в интервале от 0 до 255 (от 0x00 до 0XFF в шестнадцатеричном формате). Так как белый цвет содержит в себе все цвета, то он описывается как 0xFFFFFF, то есть каждая из трех компонент представлена максимальным значением 0xFF.

Но в ряде задач требуется указание прозрачности цвета, чтобы описать, каким образом будет выглядеть изображение, если на него наложен цвет с некоторой долей прозрачности. Для таких случаев вводится понятие альфа-канала (Alpha), который вводится как дополнительная компонента к формату RGB. Схема формата ARGB показана на рисунке.



Значения в виде ARGB обычно указываются в шестнадцатеричном формате, где каждая пара цифр представляет по порядку значения каналов Alpha, Red, Green и Blue. Например, ARGB-цвет 80FFFF00 означает желтый цвет с непрозрачностью 50.2 %. В начале идет 0x80, которое задает 50.2% значение альфа-канала, так как это 50.2% от значения 0xFF; далее первая пара FF представляет максимальное значение красной компоненты; следующая пара FF задает такую же

© 2000-2025, MetaQuotes Ltd.

силу зеленой компоненты; и последняя пара 00 представляет минимальное значение синей компоненты (отсутствие синего). Сложение зеленого и красного цвета дает желтый. Если альфа-канал не используется, запись может быть сокращена до 6 цифр RRGGBB, вот почему значения альфа-канала хранятся в верхних битах целочисленного типа uint.

В зависимости от контекста 16-ричные цифры могут записываться с префиксом '0x' или '#', например, 80FFFF00, или 0x80FFFF00, или #80FFFF00.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- зададим прозрачность
   uchar alpha=0x55;  // значение 0x55 означает 55/255=21.6 % прозрачности
   //--- выведем преобразование в ARGB для цвета clrBlue
   PrintFormat("0x%.8X - clrBlue",clrBlue);
   PrintFormat("0x%.8X - clrBlue ARGB with alpha=0x55 (transparency 21.6%%)",ColorToAR
   //--- выведем преобразование в ARGB для цвета clrGreen
   PrintFormat("0x%.8X - clrGreen",clrGreen);
   PrintFormat("0x%.8X - clrGreen ARGB with alpha=0x55 (transparency 21.6%%)",ColorToA
   //--- выведем преобразование в ARGB для цвета clrRed
   PrintFormat("0x%.8X - clrRed",clrRed);
   PrintFormat("0x%.8X - clrRed ARGB with alpha=0x55 (transparency 21.6%%)",ColorToARG
  }
```

Смотри также

Ресурсы, ResourceCreate(), TextOut(), Тип color, Типы char, short, int и long

© 2000-2025, MetaQuotes Ltd.

ColorToString

Преобразует значение цвета в строку вида "R,G,B".

```
string  ColorToString(
   color  color_value,     // значение цвета
   bool   color_name       // выводить имя цвета или нет
   );
```

Параметры

color_value

[in] Значение цвета в переменной типа color.

color_name

[in] Признак необходимости возвращать имя цвета, в случае, если значение цвета совпадает с одной из предопределенных цветовых констант.

Возвращаемое значение

Строковое представление цвета в виде "R,G,B", где R, G и B представляют собой преобразованные в строку десятичные константы со значением от 0 до 255. Если задан параметр color_name=true, то производится попытка преобразовать значение цвета к имени цвета.

Пример:

```
   string clr=ColorToString(C'0,255,0'); // зеленый цвет
   Print(clr);

   clr=ColorToString(C'0,255,0',true);   // получить цветовую константу
   Print(clr);
```

Смотри также

StringToColor, ColorToARGB

© 2000-2025, MetaQuotes Ltd.

DoubleToString

Преобразование числового значения в текстовую строку.

```
string  DoubleToString(
   double  value,      // число
   int     digits=8    // кол-во знаков после запятой
   );
```

Параметры

value

[in] Величина с плавающей точкой.

digits

[in] Формат точности. Если значение digits лежит в диапазоне от 0 до 16, то будет получено строковое представление числа с указанным количество знаков после запятой. Если значение digits лежит в диапазоне от -1 до -16, то будет получено строковое представление числа в научном формате с указанным количеством знаков после запятой. Во всех остальных случаях строковое представление числа будет содержать 8 знаков после запятой.

Возвращаемое значение

Строка, содержащая символьное представление числа в указанном формате точности.

Пример:

```
   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));
```

Смотри также

NormalizeDouble, StringToDouble

© 2000-2025, MetaQuotes Ltd.

EnumToString

Преобразование значения перечисления любого типа в текстовое представление.

```
string  EnumToString(
   any_enum  value      // значение из перечисления любого типа
   );
```

Параметры

value

[in] Значение перечисления любого типа.

Возвращаемое значение

Строка, содержащая текстовое представление значения. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция может установить в переменной _LastError следующие значения ошибок:

- ERR_INTERNAL_ERROR - ошибка среды выполнения
- ERR_NOT_ENOUGH_MEMORY - недостаточно памяти для завершения операции
- ERR_INVALID_PARAMETER - невозможно разрешить имя значения перечисления

Пример:

```
enum interval  // перечисление именованных констант
  {
   month=1,     // интервал в один месяц
   two_months,  // два месяца
   quarter,     // три месяца - квартал
   halfyear=6,  // полугодие
   year=12,     // год - 12 месяцев
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- установим временной интервал равным месяцу
   interval period=month;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- установим временной интервал равным кварталу (три месяца)
   period=quarter;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- установим временной интервал равным году (12 месяцев)
   period=year;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- проверим как выводится тип ордера
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
   Print(EnumToString(type)+"="+IntegerToString(type));

//--- проверим как выводится неверное значение
   type=WRONG_VALUE;
   Print(EnumToString(type)+"="+IntegerToString(type));

// Результат выполнения:
// month=1
// quarter=3
// year=12
// ORDER_TYPE_BUY=0
// ENUM_ORDER_TYPE::-1=-1
  }
```

Смотри также

Перечисления, Input переменные

© 2000-2025, MetaQuotes Ltd.

IntegerToString

Преобразует значение целого типа в строку указанной длины и возвращает полученную строку.

```
string  IntegerToString(
   long     number,              // число
   int      str_len=0,           // длина строки на выходе
   ushort   fill_symbol=' '      // заполнитель
   );
```

Параметры

number

[in] Число для преобразования.

str_len=0

[in] Длина строки. Если длина полученной строки окажется больше указанной, то строка не усекается. Если длина полученной строки окажется меньше, то полученная строка будет дополнена слева символом-заполнителем.

fill_symbol=' '

[in] Символ-заполнитель. По умолчанию - пробел.

Возвращаемое значение

Строка.

Пример:

```
#define DATA_TOTAL 1001

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- в цикле по DATA_TOTAL выводим строки с индексом, кратным 100
//--- в строке выводится значение индекса в четырёхсимвольном формате
//--- если длина строки меньше 4-х символов, то значение индекса цикла
//--- в строке предваряется лидирующими нулями
   for(int i=0; i<DATA_TOTAL; i++)
     {
      if(i%100==0)
         Print("Converted index value: ",IntegerToString(i,4,'0'));
     }
   /*
   результат:
   Converted index value: 0000
   Converted index value: 0100
   Converted index value: 0200
```

© 2000-2025, MetaQuotes Ltd.

```
   Converted index value: 0300
   Converted index value: 0400
   Converted index value: 0500
   Converted index value: 0600
   Converted index value: 0700
   Converted index value: 0800
   Converted index value: 0900
   Converted index value: 1000
   */
  }
```

Смотри также

StringToInteger

© 2000-2025, MetaQuotes Ltd.

ShortToString

Преобразует код символа (unicode) в односимвольную строку и возвращает полученную строку.

```
string  ShortToString(
   ushort  symbol_code      // символ
   );
```

Параметры

symbol_code

[in] Код символа. Вместо кода символа можно использовать литеральную строку, содержащую символ, либо литеральную строку с двухбайтовым шестнадцатиричным кодом, соответствующему символу из таблицы Unicode.

Возвращаемое значение

Строка.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- в цикле выведем на печать 18 символов, начиная с символа с Unicode-номером U+23E
   for(int i=0; i<18; i++)
     {
      ushort code=0x23E9+(ushort)i;
      PrintFormat("Unicode number U+%hX: %s",code,ShortToString(code));
     }
   /*
   результат:
   Unicode number U+23E9: ⏩
   Unicode number U+23EA: ⏪
   Unicode number U+23EB: ⏫
   Unicode number U+23EC: ⏬
   Unicode number U+23ED: ⏭
   Unicode number U+23EE: ⏮
   Unicode number U+23EF: ⏯
   Unicode number U+23F0: ⏰
   Unicode number U+23F1: ⏱
   Unicode number U+23F2: ⏲
   Unicode number U+23F3: ⏳
   Unicode number U+23F4: □
   Unicode number U+23F5: □
   Unicode number U+23F6: □
   Unicode number U+23F7: □
```

© 2000-2025, MetaQuotes Ltd.

```
   Unicode number U+23F8: ⏸
   Unicode number U+23F9: ⏹
   Unicode number U+23FA: ⏺
   */
  }
```

Смотри также

StringToCharArray, CharToString, StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

ShortArrayToString

Копирует часть массива в возвращаемую строку.

```
string  ShortArrayToString(
   ushort  array[],      // массив
   int     start=0,      // начальная позиция в массиве
   int     count=-1      // количество символов
   );
```

Параметры

array[]

[in] Массив типа ushort (аналог типа wchar_t).

start=0

[in] Позиция, с которой начинается копирование. По умолчанию 0.

count=-1

[in] Количество элементов массива для копирования. Определяет длину результатной строки. По умолчанию -1, что означает копирование до конца массива, либо до встречи терминального 0.

Возвращаемое значение

Строка.

Пример:

```
#define ROW_SIZE 16

ushort ExtShortArray[]={0x2190,0x2191,0x2192,0x2193,0x2194,0x2195,0x2196,0x2197,0x2198
                        0x21A0,0x21A1,0x21A2,0x21A3,0x21A4,0x21A5,0x21A6,0x21A7,0x21A8
                        0x21B0,0x21B1,0x21B2,0x21B3,0x21B4,0x21B5,0x21B6,0x21B7,0x21B8
                        0x21C0,0x21C1,0x21C2,0x21C3,0x21C4,0x21C5,0x21C6,0x21C7,0x21C8
                        0x21D0,0x21D1,0x21D2,0x21D3,0x21D4,0x21D5,0x21D6,0x21D7,0x21D8
                        0x21E0,0x21E1,0x21E2,0x21E3,0x21E4,0x21E5,0x21E6,0x21E7,0x21E8
                        0x21F0,0x21F1,0x21F2,0x21F3,0x21F4,0x21F5,0x21F6,0x21F7,0x21F8

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- рассчитаем количество строк в таблице символов размером 7x16
   int total=(int)ExtShortArray.Size()/ROW_SIZE;
//--- в цикле по строкам таблицы выведем в журнал построчно 7 наборов стрелок по 16 шт
   for(int i=0; i<total; i++)
     {
      int row=i*ROW_SIZE;
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Arrow set %u: %s",i+1,ShortArrayToString(ExtShortArray,row,ROW_SIZE
     }
   /*
   результат:
   Arrow set 1: ←↑→↓↔↕□□□□□□□□□□
   Arrow set 2: □□□□□□□□↨□□□□□□□
   Arrow set 3: □□□□□□□□□□□□□□□□
   Arrow set 4: □□□□□□□□□□□□□□□□
   Arrow set 5: □□⇒□⇔□□□□□□□□□□
   Arrow set 6: □□□□□□□□□□□□□□□□
   Arrow set 7: □□□□□□□□□□□□□□□□
   */
  }
```

Смотри также

StringToShortArray, CharArrayToString, Использование кодовой страницы

© 2000-2025, MetaQuotes Ltd.

TimeToString

Преобразование значения, содержащего время в секундах, прошедшее с 01.01.1970, в строку формата "yyyy.mm.dd hh:mi".

```
string  TimeToString(
   datetime  value,                              // число
   int       mode=TIME_DATE|TIME_MINUTES         // формат вывода
   );
```

Параметры

value

[in] Время в секундах от 00:00 1 января 1970.

mode=TIME_DATE|TIME_MINUTES

[in] Дополнительный режим вывода данных. Может быть одним или комбинированным флагом:
TIME_DATE получает результат в форме " yyyy.mm.dd " ,
TIME_MINUTES получает результат в форме " hh:mi " ,
TIME_SECONDS получает результат в форме " hh:mi:ss ".

Возвращаемое значение

Строка.

Пример:

```
datetime ExtBarTimeOpen;

//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- устанавливаем таймер в одну секунду
   EventSetTimer(1);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator deinitialization function                       |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   Comment("");
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

© 2000-2025, 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[])
  {
//--- получаем время открытия текущего бара
   ExtBarTimeOpen=time[rates_total-1];
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- запишем прошлое время открытия бара
   static datetime bar_open_time=ExtBarTimeOpen;
//--- посчитаем количество секунд, прошедших с момента открытия бара
   static int seconds=int(TimeCurrent()-ExtBarTimeOpen);
//--- если прошлое время открытия не равно текущему, значит это новый бар
//--- запишем новое время открытия как прошлое и установим секунды в ноль
   if(bar_open_time!=ExtBarTimeOpen)
     {
      bar_open_time=ExtBarTimeOpen;
      seconds=0;
     }
//--- увеличиваем и корректируем количество секунд, прошедших с момента открытия бара
   seconds++;
   if(seconds>PeriodSeconds(PERIOD_CURRENT))
      seconds=0;
//--- описание времени открытия бара в формате yyyy.mm.dd hh:mi
   string bar_time_open=TimeToString(ExtBarTimeOpen);
//--- описание текущего времени в формате yyyy.mm.dd hh:mi:ss
   string time_current=TimeToString(TimeCurrent(),TIME_DATE|TIME_MINUTES|TIME_SECONDS)
//--- количество секунд, оставшихся до открытия нового бара
   int    sec_left=PeriodSeconds(PERIOD_CURRENT)-seconds;
//--- описание количества времени, оставшегося до закрытия текущего бара в формате hh:
   string time_left=TimeToString(sec_left,TIME_MINUTES|TIME_SECONDS);
//--- создадим строку вывода
   string txt=StringFormat("Opening time of the current bar: %s\n"+
                           "Time Current: %s\n"+
                           "Seconds have passed since the bar opened: %d\n"+
                           "Approximately seconds left before bar closes: %d\n"+
```

© 2000-2025, MetaQuotes Ltd.

```
                              "Time remaining until bar closes: %s",bar_time_open,time_cu
//--- выведем в комментарии на графике время открытия бара, текущее время,
//--- количество секунд, прошедших с момента открытия текущего бара и остающихся до ег
//--- время, оставшееся до закрытия текущего бара
   Comment(txt);
   /*
   результат на M1:
   Opening time of the current bar: 2024.02.22 18:06
   Time Current: 2024.02.22 18:06:24
   Seconds have passed since the bar opened: 25
   Approximately seconds left before bar closes: 35
   Time remaining until bar closes: 00:00:35

   результат на M5:
   Opening time of the current bar: 2024.02.22 18:05
   Time Current: 2024.02.22 18:07:28
   Seconds have passed since the bar opened: 149
   Approximately seconds left before bar closes: 151
   Time remaining until bar closes: 00:02:31

   результат на H1:
   Opening time of the current bar: 2024.02.22 18:00
   Time Current: 2024.02.22 18:08:13
   Seconds have passed since the bar opened: 494
   Approximately seconds left before bar closes: 3106
   Time remaining until bar closes: 00:51:46

   результат на D1:
   Opening time of the current bar: 2024.02.22 00:00
   Time Current: 2024.02.22 18:11:01
   Seconds have passed since the bar opened: 65462
   Approximately seconds left before bar closes: 20938
   Time remaining until bar closes: 05:48:58
   */
  }
```

Смотри также

StringToTime, TimeToStruct

© 2000-2025, MetaQuotes Ltd.

NormalizeDouble

Округление числа с плавающей точкой до указанной точности.

```
double  NormalizeDouble(
   double   value,      // нормализуемое число
   int      digits      // кол-во знаков после запятой
   );
```

Параметры

value

[in] Величина с плавающей точкой.

digits

[in] Формат точности, число цифр после десятичной точки (0-8).

Возвращаемое значение

Значение типа double с заданной точностью.

Примечание

Рассчитываемые значения StopLoss, TakeProfit, а также значения цены открытия отложенных ордеров, должны быть нормализованы с точностью, значение которой можно получить функцией Digits().

Нужно иметь в виду, что нормализованное число при выводе в Журнал с помощью Print() может содержать большее количество знаков после запятой, чем вы ожидаете. Например,

```
   double a=76.671;              // нормализованное число с 3 знаками после запятой
   Print("Print(76.671)=",a);    // выведем его как есть
   Print("DoubleToString(a,8)=",DoubleToString(a,8)); // выведем с заданной точностью
```

выдаст в терминале:

```
   DoubleToString(a,8)=76.67100000
   Print(76.671)=76.67100000000001
```

Пример:

```
   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));
/*
```

© 2000-2025, MetaQuotes Ltd.

```
   Результат:
   pi= 3.1415926535897931
   NormalizeDouble(pi,3)= 3.1419999999999999
   NormalizeDouble(pi,8)= 3.1415926499999998
   NormalizeDouble(pi,0)= 3.0000000000000000
*/
```

Смотри также

DoubleToString, Вещественные типы (double, float), Приведение типов

© 2000-2025, MetaQuotes Ltd.

StringToCharArray

Посимвольно копирует преобразованную из unicode в ansi строку в указанное место массива типа uchar. Функция возвращает количество скопированных элементов.

```
int  StringToCharArray(
   string  text_string,          // строка-источник
   uchar&  array[],              // массив
   int     start=0,              // начальная позиция в массиве
   int     count=-1              // количество символов
   uint    codepage=CP_ACP       // кодовая страница
   );
```

Параметры

text_string

[in] Строка для копирования.

array[]

[out] Массив типа uchar.

start=0

[in] Позиция, с которой начинается копирование. По умолчанию 0.

count=-1

[in] Количество элементов массива для копирования. Определяет длину результатной строки. По умолчанию -1, что означает копирование до конца массива, либо до встречи терминального 0. Терминальный 0 также будет скопирован в массив-приемник, при этом размер динамического массива может быть увеличен при необходимости под размер строки. Если размер динамического массива больше длины строки, то размер массива уменьшен не будет.

codepage=CP_ACP

[in] Значение кодовой страницы. Для наиболее употребимых кодовых страниц предусмотрены соответствующие константы.

Возвращаемое значение

Количество скопированных элементов.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- string to convert
   string text = "This is a test of the StringToCharArray() function";

//--- сконвертируем входную строку в uchar-массив в соответствии с установленной кодо
   uchar char_array[];
```

© 2000-2025, MetaQuotes Ltd.

```
   int   copied=StringToCharArray(text, char_array);
   PrintFormat("String length: %u\nNumber of characters copied (with terminal 0): %d\r
               StringLen(text), copied, text);
//--- распечатаем в журнал полученный массив
   ArrayPrint(char_array, 0, " | ");
   /*
   результат:
   String length: 50
   Number of characters copied (with terminal 0): 51
   Array of characters for the string 'This is a test of the StringToCharArray() funct
   [ 0]  84 | 104 | 105 | 115 |  32 | 105 | 115 |  32 |  97 |  32 | 116 | 101 | 115 |
   [17]  32 | 116 | 104 | 101 |  32 |  83 | 116 | 114 | 105 | 110 | 103 |  84 | 111 |
   [34]  65 | 114 | 114 |  97 | 121 |  40 |  41 |  32 | 102 | 117 | 110 |  99 | 116 |
   */
  }
```

Смотри также

CharArrayToString, StringToShortArray, Использование кодовой страницы

© 2000-2025, MetaQuotes Ltd.

StringToColor

Преобразует строку типа "R,G,B" или строку, содержащую наименование цвета, в значение типа color.

```
color  StringToColor(
   string  color_string      // строковое представление цвета
   );
```

Параметры

color_string

[in] Строковое представление цвета типа "R,G,B" или название одного из предопределенных Web-цветов.

Возвращаемое значение

Значение цвета.

Пример:

```
   color str_color=StringToColor("0,127,0");
   Print(str_color);
   Print((string)str_color);
//--- немного изменим цвет
   str_color=StringToColor("0,128,0");
   Print(str_color);
   Print((string)str_color);
```

Смотри также

ColorToString, ColorToARGB

© 2000-2025, MetaQuotes Ltd.

StringToDouble

Преобразование строки, содержащей символьное представление числа, в число типа double.

```
double  StringToDouble(
   string  value      // строка
   );
```

Параметры

value

[in] Строка, содержащая символьное представление числа.

Возвращаемое значение

Значение типа double.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- string to convert
   string str = "12345.54321";
//--- конвертируем входную строку в виде вещественного числа в переменную типа double
   double converted=StringToDouble(str);
//--- выведем в журнал полученное число с точностью 8 знаков после запятой
   PrintFormat("The string '%s' is converted to the real number %.8f", str, converted)
   /*
   результат:
   The string '12345.54321' is converted to the real number 12345.54321000
   */
  }
```

Смотри также

NormalizeDouble, Вещественные типы (double, float), Приведение типов

© 2000-2025, MetaQuotes Ltd.

StringToInteger

Преобразование строки, содержащей символьное представление числа, в число типа long (целое).

```
long  StringToInteger(
   string  value      // строка
   );
```

Параметры

value

[in] Строка, содержащая число.

Возвращаемое значение

Значение типа long.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- string to convert
   string str = "12345.54321";
//--- конвертируем входную строку в виде числа в переменную типа long
   long converted=StringToInteger(str);
//--- выведем в журнал полученное число с отсечённой дробной частью
   PrintFormat("The string '%s' is converted to the integer number %I64d", str, conver
   /*
   результат:
   The string '12345.54321' is converted to the integer number 12345
   */
  }
```

Смотри также

IntegerToString, Вещественные типы (double, float), Приведение типов

© 2000-2025, MetaQuotes Ltd.

StringToShortArray

Посимвольно копирует строку в указанное место массива типа ushort. Функция возвращает количество скопированных элементов.

```
int  StringToShortArray(
   string  text_string,      // строка-источник
   ushort& array[],          // массив
   int     start=0,          // начальная позиция в массиве
   int     count=-1          // количество символов
   );
```

Параметры

text_string

[in] Строка для копирования.

array[]

[out] Массив типа ushort (аналог типа wchar_t).

start=0

[in] Позиция, с которой начинается копирование. По умолчанию 0.

count=-1

[in] Количество элементов массива для копирования. Определяет длину результатной строки. По умолчанию -1, что означает копирование до конца массива, либо до встречи терминального 0. Терминальный 0 также будет скопирован в массив-приемник, при этом размер динамического массива может быть увеличен при необходимости под размер строки. Если размер динамического массива больше длины строки, то размер массива уменьшен не будет.

Возвращаемое значение

Количество скопированных элементов.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- string with chars to convert
   string text = "Chars: ♡♫☎✈✢☏❂✉☆⌓♕♠®✧❄♣    ";
   ushort short_array[];
//--- найдём во входной строке позицию символа ":" и извлечём подстроку, начиная со сл
   int    pos=StringFind(text, ":");
   string str=(pos<0 ? text : StringSubstr(text, pos+1));
//--- удалим из полученной строки слева и справа символы перевода каретки, пробелы и с
   StringTrimLeft(str);
   StringTrimRight(str);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- скопируем полученную строку в ushort-массив и распечатаем в журнал полученный ма
   int copied=StringToShortArray(str, short_array);
   PrintFormat("String of characters length: %u\n"
               "Number of characters copied (with terminal 0): %d\n"
               "Array of chars for the string '%s':",
               StringLen(str), copied, str);
   ArrayPrint(short_array, 0, " | ");
   /*
   результат:
   String of characters length: 16
   Number of characters copied (with terminal 0): 17
   Array of chars for the string '♡♫☎✂✢☎✪✉☆⌓♛♠®✧❄♣':
   10084 |  9835 |  9742 |  9992 | 10019 |  9743 |  9728 |  9993 |  9734 |  9729 |  98
   */
  }
```

Смотри также

ShortArrayToString, StringToCharArray, Использование кодовой страницы

© 2000-2025, MetaQuotes Ltd.

StringToTime

Преобразование строки, содержащей время и/или дату в формате "yyyy.mm.dd [hh:mi]", в число типа datetime.

```
datetime  StringToTime(
   const string  time_string      // строка-дата
   );
```

Параметры

time_string

[in] Строка в одном из указанных форматов:

- "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]"

Возвращаемое значение

Значение типа datetime, содержащее количество секунд, прошедших с 01.01.1970.

Примечание

Любая последовательность символов пробела и табуляции между датой и временем считается как один пробел, чтобы не было необходимости в дополнительной обработки строки *time_string* перед вызовом StringToTime().

Пример:

```
//--- input parameters
input group    "The date can be entered in any of the formats:"
input group    "yyyy.mm.dd [hh:mi], yyyy.mm.dd [hh:mi:ss]"
input group    "yyyymmdd [hh:mi:ss], yyyymmdd [hhmiss]"
input group    "yyyy/mm/dd [hh:mi:ss], yyyy-mm-dd [hh:mi:ss]"
input string   InpDateStr;    // Please enter the date here as a string

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- преобразуем введённое во входных параметрах время в виде строки в значение datet
   datetime time=StringToTime(InpDateStr);
//--- выводим в журнал введённую строку и полученное время
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("Date entered as a string in the form '%s' is converted to datetime in
               InpDateStr, TimeToString(time, TIME_DATE|TIME_MINUTES|TIME_SECONDS));
//--- создаём вертикальную линию на полученной дате-времени и смещаем график в это ра
   if(CreateVLine(time))
      ChartNavigateToTime(time);
   /*
   результат:
   Date entered as a string in the form '' is converted to datetime in the form '1970.
   Date entered as a string in the form '2024' is converted to datetime in the form '2
   Date entered as a string in the form '202400' is converted to datetime in the form
   Date entered as a string in the form '20240000' is converted to datetime in the fo
   Date entered as a string in the form '2024022410' is converted to datetime in the
   Date entered as a string in the form '20240224 10' is converted to datetime in the
   Date entered as a string in the form '20240224 01' is converted to datetime in the
   Date entered as a string in the form '20240224 0030' is converted to datetime in t
   Date entered as a string in the form '20240224 0100' is converted to datetime in t
   */
  }
//+------------------------------------------------------------------+
//| Создаёт объект вертикальную линию                                |
//+------------------------------------------------------------------+
bool CreateVLine(const datetime line_time)
  {
   ResetLastError();

   string name=MQLInfoString(MQL_PROGRAM_NAME)+"_VLINE";
   if(!ObjectCreate(0, name, OBJ_VLINE, 0, line_time, 0))
     {
      Print("ObjectCreate() failed. Error code: ", GetLastError());
      return(false);
     }
   ObjectSetInteger(0, name, OBJPROP_STYLE, STYLE_DOT);
   ObjectSetInteger(0, name, OBJPROP_SELECTABLE, true);

   return(true);
  }
//+------------------------------------------------------------------+
//| Смещает график на указанное время открытия бара                  |
//+------------------------------------------------------------------+
bool ChartNavigateToTime(const datetime time)
  {
   ChartSetInteger(0, CHART_AUTOSCROLL, false);
   ResetLastError();

   int bar=iBarShift(_Symbol, PERIOD_CURRENT, time);
   if(bar<0)
     {
      PrintFormat("%s: iBarShift() failed. Error code: %d", __FUNCTION__, GetLastErro
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }

   long first=0;
   if(!ChartGetInteger(0, CHART_FIRST_VISIBLE_BAR, 0, first))
      {
       PrintFormat("%s: ChartGetInteger() failed. Error code: %d", __FUNCTION__, GetLas
       return(false);
      }

   return(ChartNavigate(0, CHART_CURRENT_POS, (int)first-bar));
  }
```

Смотри также

TimeToString, TimeToStruct

© 2000-2025, MetaQuotes Ltd.

StringFormat

Форматирует полученные параметры и возвращает строку.

```
string  StringFormat(
   string  format,      // строка с описанием формата
   ...     ...          // параметры
   );
```

Параметры

format

[in] Строка, содержащая способ форматирования. Правила форматирования такие же, как и для функции PrintFormat

...

[in] Параметры, разделенные запятой.

Возвращаемое значение

Строка.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- строковые переменные
   string output_string;
   string temp_string;
   string format_string;
//--- подготовим заголовок спецификации
   temp_string=StringFormat("Спецификация контракта для %s:\n",_Symbol);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int digits=(int)SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   temp_string=StringFormat("   SYMBOL_DIGITS = %d (количество знаков после запятой)\n
                            digits);
   StringAdd(output_string,temp_string);
//--- вывод значения double с переменным количеством цифр после десятичной точки
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   format_string=StringFormat("   SYMBOL_POINT = %%.%df (значение одного пункта)\n",
                              digits);
   temp_string=StringFormat(format_string,point_value);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   temp_string=StringFormat("   SYMBOL_SPREAD = %d (текущий спред в пунктах)\n",
                            spread);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int min_stop=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_STOPS_LEVEL = %d (минимальный отступ в пу
                            min_stop);
   StringAdd(output_string,temp_string);
//--- вывод значения double без дробной части
   double contract_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_CONTRACT_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_CONTRACT_SIZE = %.f (размер контракта)\n"
                            contract_size);
   StringAdd(output_string,temp_string);
//--- вывод значения double с точностью по умолчанию
   double tick_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_TICK_SIZE = %f (минимальное изменение це
                            tick_size);
   StringAdd(output_string,temp_string);
//--- определение способа начисления свопов
   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 (нет сп
      case SYMBOL_SWAP_MODE_POINTS: str_swap_mode="SYMBOL_SWAP_MODE_POINTS (в пунктах)
      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 (пе
     }
//--- вывод значения string
   temp_string=StringFormat("   SYMBOL_SWAP_MODE = %s\n",
                            str_swap_mode);
   StringAdd(output_string,temp_string);
//--- вывод значения double с точностью по умолчанию
   double swap_long=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_LONG);
```

© 2000-2025, MetaQuotes Ltd.

```
   temp_string=StringFormat("   SYMBOL_SWAP_LONG = %f (своп на покупку)\n",
                            swap_long);
   StringAdd(output_string,temp_string);
//--- вывод значения double с точностью по умолчанию
   double swap_short=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_SHORT);
   temp_string=StringFormat("   SYMBOL_SWAP_SHORT = %f (своп на продажу)\n",
                            swap_short);
   StringAdd(output_string,temp_string);
//--- определение режима торговли
   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 (тор
      case SYMBOL_TRADE_MODE_LONGONLY: str_trade_mode="SYMBOL_TRADE_MODE_LONGONLY (раз
      case SYMBOL_TRADE_MODE_SHORTONLY: str_trade_mode="SYMBOL_TRADE_MODE_SHORTONLY (р
      case SYMBOL_TRADE_MODE_CLOSEONLY: str_trade_mode="SYMBOL_TRADE_MODE_CLOSEONLY (р
      case SYMBOL_TRADE_MODE_FULL: str_trade_mode="SYMBOL_TRADE_MODE_FULL (нет огранич
     }
//--- вывод значения string
   temp_string=StringFormat("   SYMBOL_TRADE_MODE = %s\n",
                            str_trade_mode);
   StringAdd(output_string,temp_string);
//--- вывод значения double в компактном виде
   double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   temp_string=StringFormat("   SYMBOL_VOLUME_MIN = %g (минимальный объем сделки)\n",v
   StringAdd(output_string,temp_string);
//--- вывод значения double в компактном виде
   double volume_step=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_STEP);
   temp_string=StringFormat("   SYMBOL_VOLUME_STEP = %g (минимальный шаг изменения об
   StringAdd(output_string,temp_string);
//--- вывод значения double в компактном виде
   double volume_max=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MAX);
   temp_string=StringFormat("   SYMBOL_VOLUME_MAX = %g (максимальный объем сделки)\n",
   StringAdd(output_string,temp_string);
//--- определение способа вычисления величины залоговых средств
   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 (фьючерсы)
      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 на
      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 (т
      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
     }
//--- вывод значения string
   temp_string=StringFormat("   SYMBOL_TRADE_CALC_MODE = %s\n",
                            str_calc_mode);
   StringAdd(output_string,temp_string);
//--- вывод значения double с 2 цифрами после десятичной точки
   double margin_initial=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_INITIAL);
   temp_string=StringFormat("   SYMBOL_MARGIN_INITIAL = %.2f (начальная маржа)\n",
                            margin_initial);
   StringAdd(output_string,temp_string);
//--- вывод значения double с 2 цифрами после десятичной точки
   double margin_maintenance=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_MAINTENANCE);
   temp_string=StringFormat("   SYMBOL_MARGIN_MAINTENANCE = %.2f (поддерживающая марж
```

© 2000-2025, MetaQuotes Ltd.

```
                              margin_maintenance);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int freeze_level=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_FREEZE_LEVEL = %d (дистанция заморозки ор
                              freeze_level);
   StringAdd(output_string,temp_string);
   Print(output_string);
   Comment(output_string);
/* результат выполнения
   Спецификация контракта для EURUSD:
     SYMBOL_DIGITS = 5 (количество знаков после запятой)
     SYMBOL_POINT = 0.00001 (значение одного пункта)
     SYMBOL_SPREAD = 10 (текущий спред в пунктах)
     SYMBOL_TRADE_STOPS_LEVEL = 18 (минимальный отступ в пунктах для стоп-ордеров)
     SYMBOL_TRADE_CONTRACT_SIZE = 100000 (размер контракта)
     SYMBOL_TRADE_TICK_SIZE = 0.000010 (минимальное изменение цены)
     SYMBOL_SWAP_MODE = SYMBOL_SWAP_MODE_POINTS (в пунктах)
     SYMBOL_SWAP_LONG = -0.700000 (своп на покупку)
     SYMBOL_SWAP_SHORT = -1.000000 (своп на продажу)
     SYMBOL_TRADE_MODE = SYMBOL_TRADE_MODE_FULL (нет ограничений на торговые операции)
     SYMBOL_VOLUME_MIN = 0.01 (минимальный объем сделки)
     SYMBOL_VOLUME_STEP = 0.01 (минимальный шаг изменения объема сделки)
     SYMBOL_VOLUME_MAX = 500 (максимальный объем сделки)
     SYMBOL_TRADE_CALC_MODE = SYMBOL_CALC_MODE_FOREX (Forex)
     SYMBOL_MARGIN_INITIAL = 0.00 (начальная маржа)
     SYMBOL_MARGIN_MAINTENANCE = 0.00 (поддерживающая маржа)
     SYMBOL_TRADE_FREEZE_LEVEL = 0 (дистанция заморозки операций в пунктах)
*/
  }
```

Смотри также

PrintFormat, DoubleToString,ColorToString, TimeToString

© 2000-2025, MetaQuotes Ltd.

Математические функции

Набор математических и тригонометрических функций.

Математические функции изначально были предназначены для проведения математических операций над скалярными величинами. Теперь большая часть этих функций может использоваться и с новыми типами данных — матрицами и векторами — MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh. В этом случае матрица или вектор обрабатываются почленно. Пример:

```
//---
  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]]
  */
```

В случае MathMod и MathPow в качестве второго параметра может быть использован как скаляр, так и матрица или вектор соответствующего размера.

Функция	Действие
MathAbs	Возвращает абсолютное значение (значение по модулю) переданного ей числа
MathArccos	Возвращает значение арккосинуса x в радианах
MathArcsin	Возвращает значение арксинуса x в радианах
MathArctan	Возвращает арктангенс x в радианах
MathArctan2	Возвращает в радианах значение угла, тангенс которого равен отношению двух указанных чисел
MathClassify	Возвращает тип вещественного числа
MathCeil	Возвращает ближайшее сверху целое числовое значение
MathCos	Возвращает косинус числа
MathExp	Возвращает экспоненту числа
MathFloor	Возвращает ближайшее снизу целое числовое значение
MathLog	Возвращает натуральный логарифм

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
MathLog10	Возвращает логарифм числа по основанию 10
MathMax	Возвращает максимальное из двух числовых значений
MathMin	Возвращает минимальное из двух числовых значений
MathMod	Возвращает вещественный остаток от деления двух чисел
MathPow	Возводит основание в указанную степень
MathRand	Возвращает псевдослучайное целое число в диапазоне от 0 до 32767
MathRound	Округляет число до ближайшего целого
MathSin	Возвращает синус числа
MathSqrt	Возвращает квадратный корень
MathSrand	Устанавливает начальное состояние генератора псевдослучайных целых чисел
MathTan	Возвращает тангенс числа
MathIsValidNumber	Проверяет корректность действительного числа
MathExpm1	Возвращает значение выражения MathExp(x)-1
MathLog1p	Возвращает значение выражения MathLog(1+x)
MathArccosh	Возвращает значение гиперболического арккосинуса
MathArcsinh	Возвращает значение гиперболического арксинуса
MathArctanh	Возвращает значение гиперболического арктангенса
MathCosh	Возвращает гиперболический косинус
MathSinh	Возвращает гиперболический синус
MathTanh	Возвращает гиперболический тангенс
MathSwap	Меняет порядок байтов в значении типа ushort/uint/ulong

© 2000-2025, MetaQuotes Ltd.

MathAbs

Возвращает абсолютное значение (значение по модулю) переданного ей числа.

```
double  MathAbs(
   double  value      // число
   );
```

Параметры

value

[in] Числовая величина.

Возвращаемое значение

Значение типа double, больше или равно нулю.

Примечание

Вместо функции MathAbs() можно использовать функцию fabs().

Пример:

```
//--- input parameters
input int      InpValue = -10;   // Enter any value here

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем абсолютное значение введённого во входных параметрах числа
   uint abs_value=MathAbs(InpValue);
//--- распечатываем значения в журнал
   PrintFormat("The entered value %d received the value %u after using the MathAbs() f
  }
```

© 2000-2025, MetaQuotes Ltd.

MathArccos

Возвращает значение арккосинуса x в диапазоне 0 к π в радианах.

```
double  MathArccos(
   double  value     // -1<val<1
   );
```

Параметры

value

[in] Значение value между -1 и 1, арккосинус которого должен быть вычислен.

Возвращаемое значение

Арккосинус числа в радианах. Если value меньше -1 или больше 1, функция возвращает NaN (неопределенное значение).

Примечание

Вместо функции MathArccos() можно использовать функцию acos().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/1000);
   delta[0]=-1;
//--- получим 101 значение от -1 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение арккосинуса для каждого значения вектора X
   vector Y=MathArccos(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

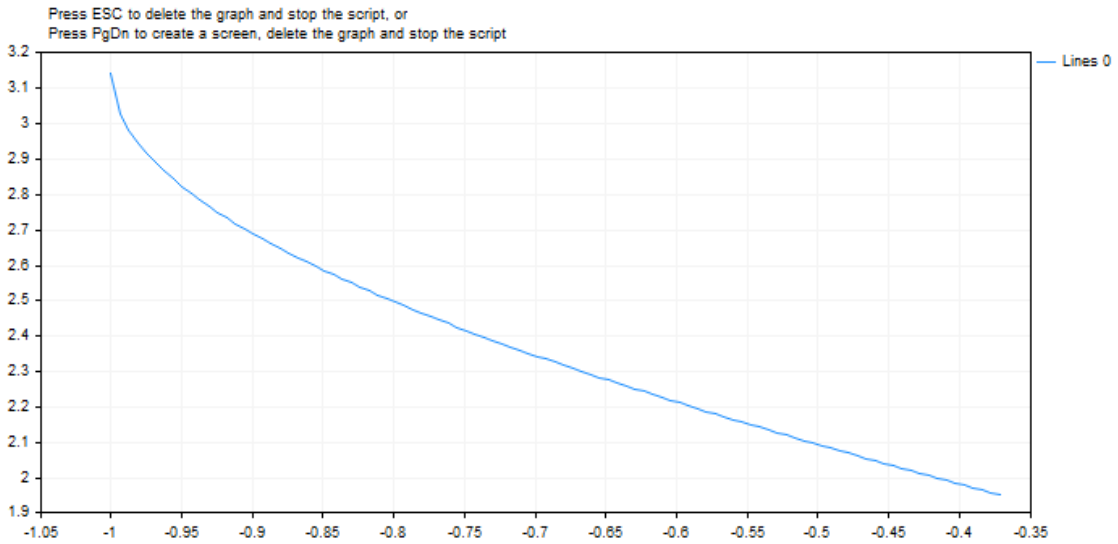
© 2000-2025, MetaQuotes Ltd.

```
//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArcsin

Возвращает арксинус x в диапазоне от -π/2 до π/2 радианов.

```
double  MathArcsin(
   double  value      // -1<value<1
   );
```

Параметры

value

[in] Значение value между -1 и 1, арксинус которого должен быть вычислен.

Возвращаемое значение

Арксинус числа value в радианах в диапазоне от -π/2 до π/2 радианов. Если value меньше -1 или больше 1, функция возвращает NaN (неопределенное значение).

Примечание

Вместо функции MathArcsin() можно использовать функцию asin().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/1000);
   delta[0]=-1;
//--- получим 101 значение от -1 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение синуса для каждого значения вектора X
   vector Y=MathArcsin(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

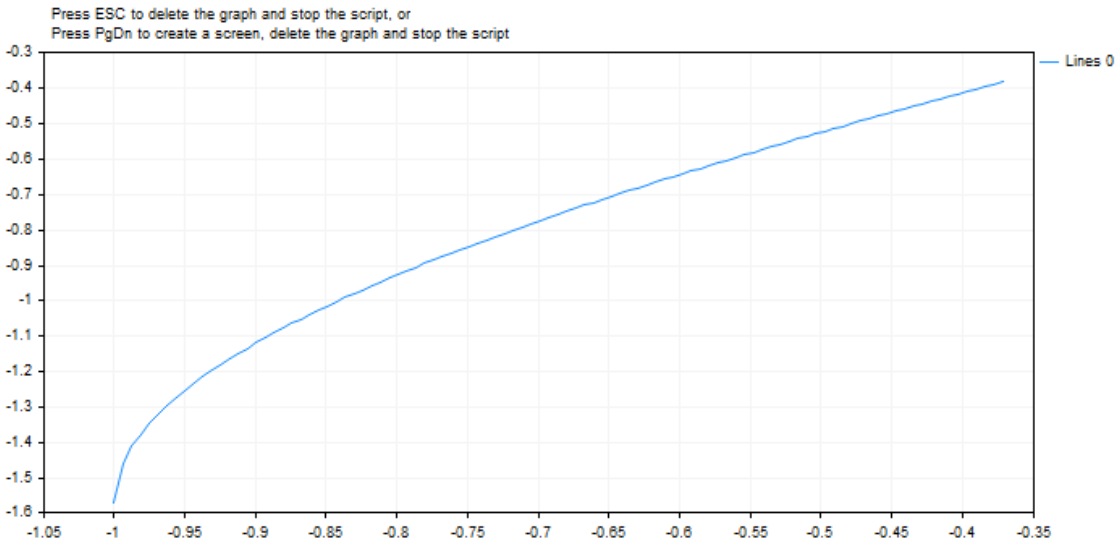
© 2000-2025, MetaQuotes Ltd.

```
//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArctan

Возвращает арктангенс x. Если x равен 0, функция возвращает 0.

```
double  MathArctan(
   double  value      // тангенс
   );
```

Параметры

value

[in] Число, представляющее тангенс.

Возвращаемое значение

MathArctan возвращает значение в диапазоне от -π/2 до π/2 радианов.

Примечание

Вместо функции MathArctan() можно использовать функцию atan().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- получим 101 значение от 0 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение арктангенса для каждого значения вектора X
   vector Y=MathArctan(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

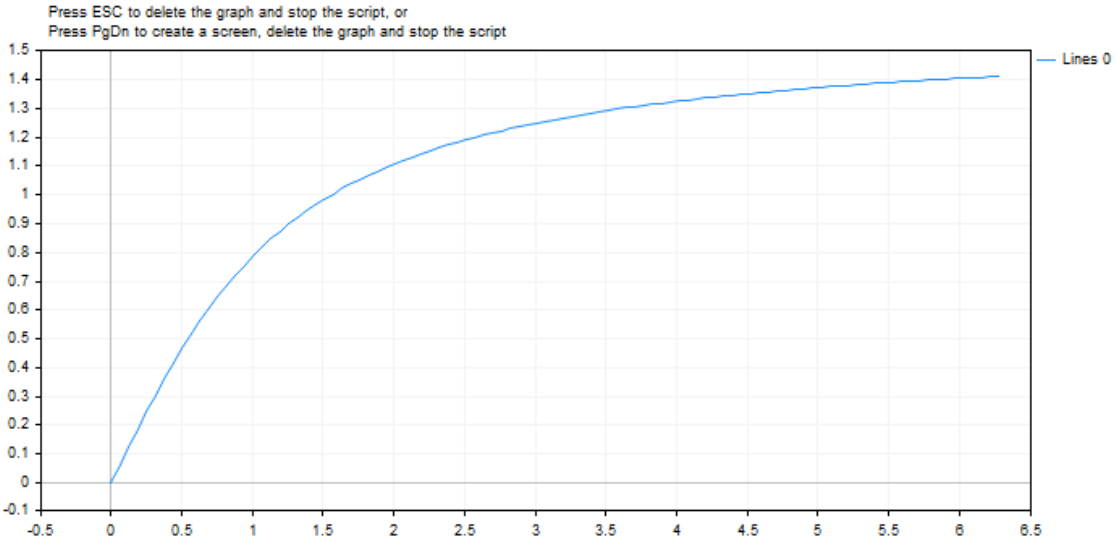
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



© 2000-2025, MetaQuotes Ltd.

MathArctan2

Возвращает в радианах значение угла, тангенс которого равен отношению двух указанных чисел.

```
double  MathArctan2(
   double  y      // y-координата точки
   double  x      // x-координата точки
   );
```

Параметры

y

[in] Число, представляющее координату Y.

x

[in] Число, представляющее координату X.

Возвращаемое значение

MathArctan2 возвращает значение угла θ в диапазоне от -π до π радиан, так что MathTan(θ) =y/x.

Обратите внимание на следующее:

- Для (x, y) в квадранте 1, 0 < θ < π/2
- Для (x, y) в квадранте 2, π/2 < θ≤π
- Для (x, y) в квадранте 3, -π < θ < -π/2
- Для (x, y) в квадранте 4, -π/2 < θ < 0

Для точек за пределами указанных квадрантов возвращаемое значение указано ниже:

- Если y равно 0, и x не является отрицательным, то θ = 0.
- Если y равно 0, и x является отрицательным, то θ = π.
- Если y — положительное число, а x равно 0, то θ = π/2.
- Если y — отрицательное число, а x равно 0, то θ = -π/2.
- Если y равен 0, и x равен 0, то θ = 0.

Примечание

Вместо функции MathArctan2() можно использовать функцию atan2().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

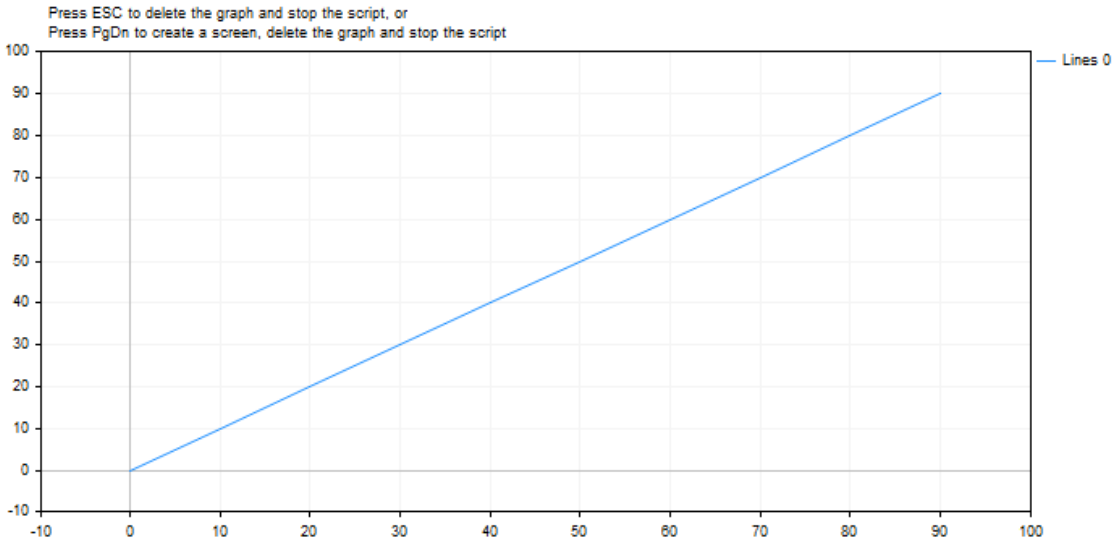
```
void OnStart()
  {
   vector delta=vector::Full(10,10);
   delta[0]=0;
//--- получим 101 значение от 0 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение арктангенса для каждого значения вектора X
   vector Y=delta.CumSum();

   Print("vector delta = \n",delta);
   Print("vector X = \n",X);
   Print("vector Y = \n",Y);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[];;
   double y_array[];;
   X.Swap(x_array);
   Y.Swap(y_array);

   double array[10];
   for(int i=0; i<10; i++)
     {
      array[i]=MathArctan2(y_array[i],x_array[i]);
     }

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:

© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

MathClassify

Определяет тип вещественного числа и возвращает результат в виде значения из перечисления ENUM_FP_CLASS

```
ENUM_FP_CLASS  MathClassify(
   double  value      // вещественное число
   );
```

Параметры

value

[in] Проверяемое вещественное число

Возвращаемое значение

Значение из перечисления ENUM_FP_CLASS

ENUM_FP_CLASS

Идентификатор	Описание
FP_SUBNORMAL	Субнормальное число, которое находится ближе к нулю, чем наименьшее представимое нормализованное число DBL_MIN (2.2250738585072014e-308)
FP_NORMAL	Нормализованное число, которое находится в диапазоне от 2.2250738585072014e-308 до 1.7976931348623158e+308
FP_ZERO	Положительный или отрицательный ноль
FP_INFINITE	Число, которое не может быть представлено соответствующим типом — положительная или отрицательная бесконечность
FP_NAN	Не является числом

Пример:

```
//+------------------------------------------------------------------+
//| 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));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- test infinity
  double inf=double("inf");
  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));
 }
 /*
 Результат:
   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
*/
//+------------------------------------------------------------------+
```

Смотри также

Вещественные типы (double, float), MathIsValidNumber

© 2000-2025, MetaQuotes Ltd.

MathCeil

Возвращает ближайшее сверху целое числовое значение.

```
double  MathCeil(
   double  val      // число
   );
```

Параметры

val

[in] Числовая величина.

Возвращаемое значение

Числовое значение, представляющую наименьшее целое число, которое больше или равно val.

Примечание

Вместо функции MathCeil() можно использовать функцию ceil().

Пример:

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявляем переменные для преобразования
   double value=0;                     // вещественное число для MathCeil-преобразова
   int    ceil_value=0;                // сюда получаем результат
//--- в цикле по количеству десятичных приращений вещественного числа
   for(int i=0; i<VALUES_TOTAL; i++)
     {
      //--- увеличиваем значение числа,
      //--- получаем ближайшее сверху целое числовое значение
      //--- и выводим в журнал контрольные значения
      value+=0.1;
      ceil_value=(int)MathCeil(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, ceil value: %d",value,ceil_value);
      /*
      результат:
      value: 0.1, ceil value: 1
      value: 0.2, ceil value: 1
      value: 0.3, ceil value: 1
      value: 0.4, ceil value: 1
      value: 0.5, ceil value: 1
      value: 0.6, ceil value: 1
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, ceil value: 1
      value: 0.8, ceil value: 1
      value: 0.9, ceil value: 1
      value: 1.0, ceil value: 1
      value: 1.1, ceil value: 2
      value: 1.2, ceil value: 2
      value: 1.3, ceil value: 2
      value: 1.4, ceil value: 2
      value: 1.5, ceil value: 2
      value: 1.6, ceil value: 2
      value: 1.7, ceil value: 2
      value: 1.8, ceil value: 2
      value: 1.9, ceil value: 2
      value: 2.0, ceil value: 2
      value: 2.1, ceil value: 3
      value: 2.2, ceil value: 3
      value: 2.3, ceil value: 3
      value: 2.4, ceil value: 3
      value: 2.5, ceil value: 3
      value: 2.6, ceil value: 3
      value: 2.7, ceil value: 3
      value: 2.8, ceil value: 3
      value: 2.9, ceil value: 3
      value: 3.0, ceil value: 3
      value: 3.1, ceil value: 4
      */
   }
}
```

© 2000-2025, MetaQuotes Ltd.

MathCos

Функция возвращает косинус угла.

```
double  MathCos(
   double  value      // число
   );
```

Параметры

value

[in] Угол в радианах.

Возвращаемое значение

Значение типа double в диапазоне от -1 до 1.

Примечание

Вместо функции MathCos() можно использовать функцию cos().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- получим 101 значение от 0 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение косинуса для каждого значения вектора X
   vector Y=MathCos(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

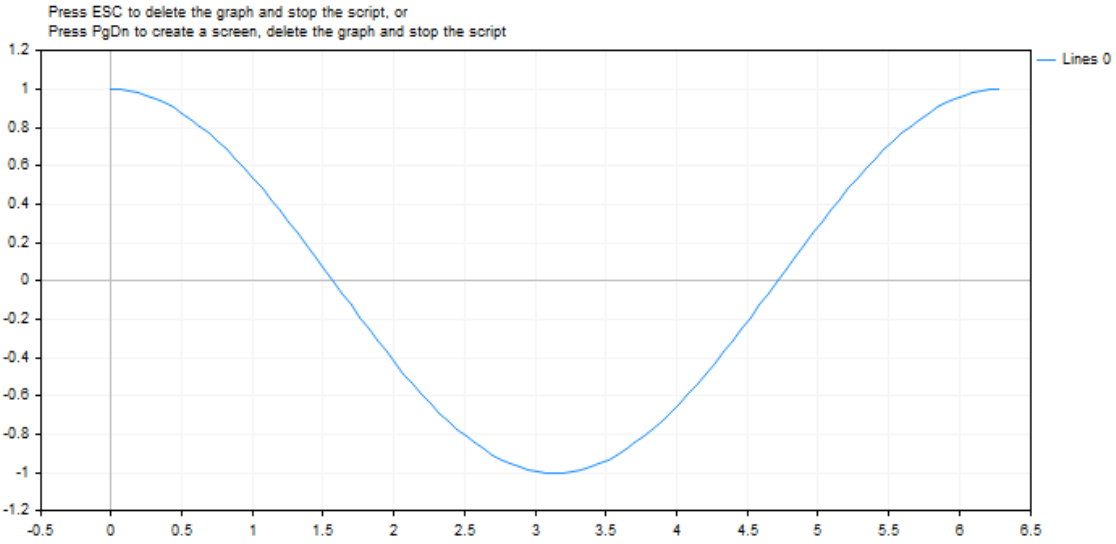
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



© 2000-2025, MetaQuotes Ltd.

MathExp

Возвращает значение числа e в степени d.

```
double  MathExp(
   double  value      // степень для числа e
   );
```

Параметры

value

[in] Число, определяющее степень.

Возвращаемое значение

Число типа double. При переполнении функция возвращает INF (бесконечность), в случае потери порядка точности MathExp возвращает 0.

Примечание

Вместо функции MathExp() можно использовать функцию exp().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим 9 значений от 0 до 8 с шагом 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- рассчитываем значение числа "e"(число Эйлера) в степени каждого значения вектора
   X=MathExp(X);
   Print("MathExp(X) = \n",X);

//--- переносим рассчитанные значения из вектора в массив
   double y_array[];
   X.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

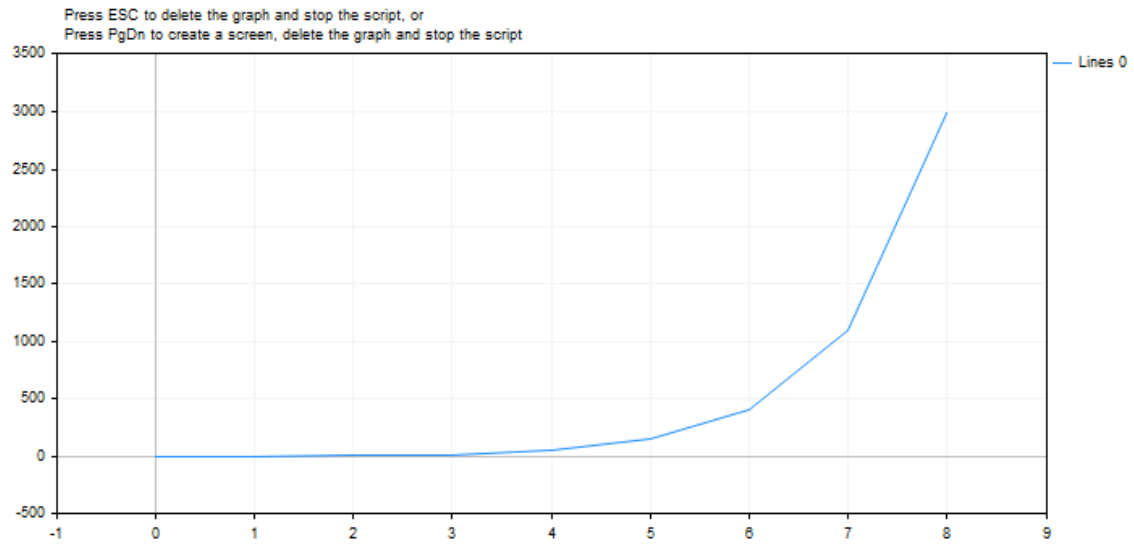
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Заполняет вектор значением value с приращением step              |
//+------------------------------------------------------------------+
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;
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathFloor

Возвращает ближайшее снизу целое числовое значение.

```
double  MathFloor(
   double  value     // число
   );
```

Параметры

val

[in] Числовое значение.

Возвращаемое значение

Числовое значение, представляющее наибольшее целое число, которое меньше или равно value.

Примечание

Вместо функции MathFloor() можно использовать функцию floor().

Пример:

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявляем переменные для преобразования
   double value=0;                     // вещественное число для MathFloor-преобразова
   int    floor_value=0;               // сюда получаем результат
//--- в цикле по количеству десятичных приращений вещественного числа
   for(int i=0; i<VALUES_TOTAL; i++)
     {
      //--- увеличиваем значение числа,
      //--- получаем ближайшее снизу целое числовое значение
      //--- и выводим в журнал контрольные значения
      value+=0.1;
      floor_value=(int)MathFloor(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, floor value: %d",value,floor_value);
      /*
      результат:
      value: 0.1, floor value: 0
      value: 0.2, floor value: 0
      value: 0.3, floor value: 0
      value: 0.4, floor value: 0
      value: 0.5, floor value: 0
      value: 0.6, floor value: 0
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, floor value: 0
      value: 0.8, floor value: 0
      value: 0.9, floor value: 0
      value: 1.0, floor value: 1
      value: 1.1, floor value: 1
      value: 1.2, floor value: 1
      value: 1.3, floor value: 1
      value: 1.4, floor value: 1
      value: 1.5, floor value: 1
      value: 1.6, floor value: 1
      value: 1.7, floor value: 1
      value: 1.8, floor value: 1
      value: 1.9, floor value: 1
      value: 2.0, floor value: 2
      value: 2.1, floor value: 2
      value: 2.2, floor value: 2
      value: 2.3, floor value: 2
      value: 2.4, floor value: 2
      value: 2.5, floor value: 2
      value: 2.6, floor value: 2
      value: 2.7, floor value: 2
      value: 2.8, floor value: 2
      value: 2.9, floor value: 2
      value: 3.0, floor value: 3
      value: 3.1, floor value: 3
      */
     }
```

© 2000-2025, MetaQuotes Ltd.

MathLog

Возвращает натуральный логарифм.

```
double  MathLog(
   double  value      // число для взятия логарифма
   );
```

Параметры

val

[in] Значение, логарифм которого должен быть вычислен.

Возвращаемое значение

Натуральный логарифм value в случае успеха. Если значение val отрицательно, функция возвращает NaN (неопределенное значение). Если value равно 0, функция возвращает INF (бесконечность) .

Примечание

Вместо функции MathLog() можно использовать функцию log().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим 9 значений от 0 до 8 с шагом 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- рассчитываем логарифм для каждого значения вектора X
   X=MathLog(X);
   Print("MathLog(X) = \n",X);

//--- переносим рассчитанные значения из вектора в массив
   double y_array[];
   X.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(y_array,clrDodgerBlue);
```

© 2000-2025, MetaQuotes Ltd.

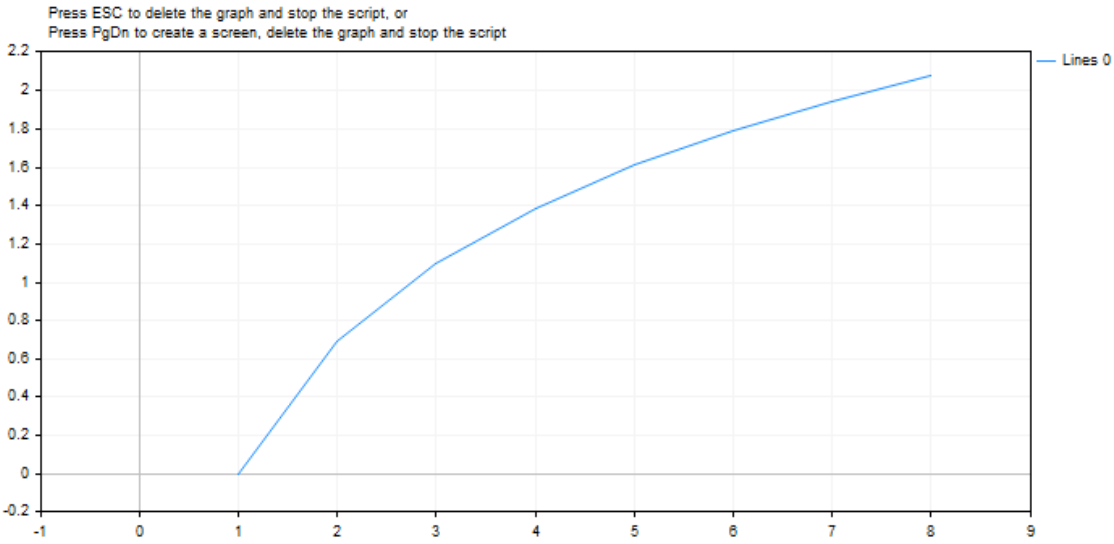
```
//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
   /*
   результат:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog(X) =
   [-inf,0,0.6931471805599453,1.09861228866811,1.386294361119891,1.6094379124341,1.791
   */
  }
//+------------------------------------------------------------------+
//| Заполняет вектор значением value с приращением step              |
//+------------------------------------------------------------------+
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;
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:

© 2000-2025, MetaQuotes Ltd.



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathLog

Возвращает логарифм числа по основанию 10.

```
double  MathLog10(
   double  val      // число для взятия логарифма
   );
```

Параметры

val

[in] Значение, десятичный логарифм которого должен быть вычислен.

Возвращаемое значение

Десятичный логарифм val в случае успеха. Если значение val отрицательно, функция возвращает NaN (неопределенное значение). Если val равно 0, функция возвращает INF (бесконечность) .

Примечание

Вместо функции MathLog10() можно использовать функцию log10().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим 9 значений от 0 до 8 с шагом 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- рассчитываем десятичный логарифм для каждого значения вектора X
   X=MathLog10(X);
   Print("MathLog10(X) = \n",X);

//--- переносим рассчитанные значения из вектора в массив
   double y_array[];
   X.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

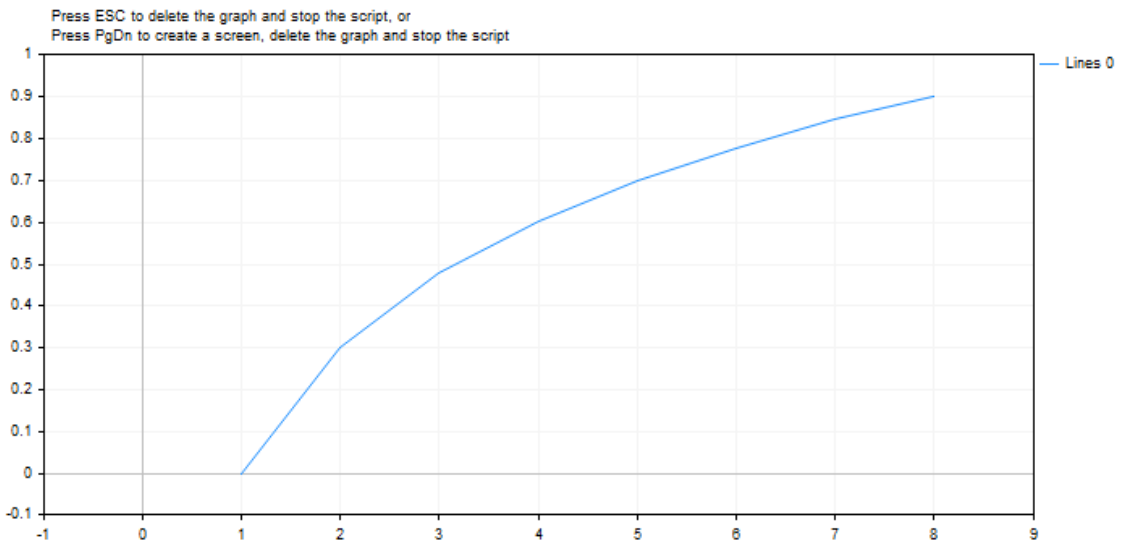
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
   /*
   результат:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog10(X) =
   [-inf,0,0.3010299956639812,0.4771212547196624,0.6020599913279624,0.6989700043360189
   */
  }
//+------------------------------------------------------------------+
//| Заполняет вектор значением value с приращением step              |
//+------------------------------------------------------------------+
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;
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



© 2000-2025, MetaQuotes Ltd.

Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathMax

Функция возвращает максимальное из двух числовых значений.

```
double  MathMax(
   double  value1,      // первое число
   double  value2       // второе число
   );
```

Параметры

value1

[in] Первое числовое значение.

value2

[in] Второе числовое значение.

Возвращаемое значение

Большее из двух чисел.

Примечание

Вместо функции MathMax() можно использовать функцию fmax(). Функции fmax(), fmin(), MathMax(), MathMin() могут работать с целыми типами без преобразования к типу double.

Если в функцию передаются параметры разных типов, то параметр младшего типа автоматически приводится к старшему типу. Тип возвращаемого значения соответствует старшему типу.

Если передаются данные одного типа, то никакого преобразования не производится.

Пример:

```
//--- input parameters
input int                  InpPeriod = 10;            // Период расчёта скользящей сре
input ENUM_MA_METHOD       InpMethod = MODE_SMA;      // Метод расчёта скользящей сред
input ENUM_APPLIED_PRICE   InpPrice  = PRICE_CLOSE;   // Цена расчёта скользящей средн

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- если для периода скользящей средней задано значение меньше 1, то использоваться
   int period=(InpPeriod<1 ? 10 : InpPeriod);
//--- создаём хэндл индикатора Moving Average
   int handle=iMA(Symbol(),Period(),period,0,InpMethod,InpPrice);
   if(handle==INVALID_HANDLE)
     {
      Print("Не удалось создать хэндл индикатора Moving Average. Ошибка ",GetLastError
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- получаем текущую цену Bid
   double bid=0;
   ResetLastError();
   if(!SymbolInfoDouble(Symbol(),SYMBOL_BID,bid))
     {
      Print("Не удалось получить цену Bid. Ошибка ",GetLastError());
      return;
     }
//--- получаем значение скользящей средней на текущем баре
   double array[1];
   int    copied=CopyBuffer(handle,0,0,1,array);
   if(copied!=1)
     {
      Print("Не удалось получить данные Moving Average. Ошибка ",GetLastError());
      return;
     }
//--- получаем наибольшую цену из двух (цена Bid и значение Moving Average) и выводим
   double max_price=MathMax(bid,array[0]);
   PrintFormat("Bid: %.*f, Moving Average: %.*f, наибольшая цена из двух: %.*f",_Digit
   PrintFormat("Цена Bid %s скользящей средней",(bid>array[0] ? "выше" : bid<array[0]
  }
```

© 2000-2025, MetaQuotes Ltd.

MathMin

Функция возвращает минимальное из двух числовых значений.

```
double  MathMin(
   double  value1,      // первое число
   double  value2       // второе число
   );
```

Параметры

value1

[in] Первое числовое значение.

value2

[in] Второе числовое значение.

Возвращаемое значение

Меньшее из двух чисел.

Примечание

Вместо функции MathMin() можно использовать функцию fmin(). Функции fmax(), fmin(), MathMax(), MathMin() могут работать с целыми типами без преобразования к типу double.

Если в функцию передаются параметры разных типов, то параметр младшего типа автоматически приводится к старшему типу. Тип возвращаемого значения соответствует старшему типу.

Если передаются данные одного типа, то никакого преобразования не производится.

Пример:

```
//--- input parameters
input uint  InpBars  =  100000;  // Желаемое количество баров

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим количество доступных баров на сервере
   uint bars_total = Bars(Symbol(),Period());
   if(bars_total==0)
     {
      PrintFormat("Данные для таймсерии %s %s еще не сформированы в терминале. Повтори
      return;
     }
//--- получим минимальное количество баров из двух значений: из доступных на сервере и
   int bars = (int)MathMin(bars_total,InpBars);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если запрошено баров больше, чем доступно на сервере - сообщим об этом в журнале
   if(bars_total<InpBars)
      PrintFormat("Количество баров на сервере (%u) меньше, чем было запрошено (%u)",b
//--- выведем в журнал доступное для работы количество баров
   Print("Доступно баров для работы: ",bars);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathMod

Возвращает вещественный остаток от деления двух чисел.

```
double  MathMod(
   double  value,       // делимое
   double  value2       // делитель
   );
```

Параметры

value

[in] Значение делимого.

value2

[in] Значение делителя.

Возвращаемое значение

Функция MathMod рассчитывает вещественный остаток f от val / y таким образом, что val = i * y + f , где i является целым числом, f имеет тот же знак, что и val, и абсолютное значение f меньше, чем абсолютное значение y.

Примечание

Вместо функции MathMod() можно использовать функцию fmod().

Пример:

```
#property script_show_inputs

//--- input parameters
input double   InpDividentValue  =  10;   // Dividend value
input double   InpDivisorValue   =  3;    // Divisor value

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем вещественный остаток от деления введённых во входных параметрах чисел
   double res=MathMod(InpDividentValue,InpDivisorValue);
//--- распечатываем результат в журнал
   PrintFormat("Real remainder when dividing %.2f by %.2f = %.2f",InpDividentValue,Inp
  }
```

© 2000-2025, MetaQuotes Ltd.

MathPow

Возводит основание в указанную степень.

```
double  MathPow(
   double  base,          // основание
   double  exponent       // показатель степени
   );
```

Параметры

base

[in] Основание.

exponent

[in] Значение степени.

Возвращаемое значение

Значение основания, возведенного в указанную степень.

Примечание

Вместо функции MathPow() можно использовать функцию pow().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#property script_show_inputs

#include <Graphics\Graphic.mqh>

//--- input parameters
input double   InpExponentValue  =  2;     // exponent value

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим 11 значений от 0 до 10 с шагом 1
   vector X(11,VectorArange);
   Print("vector X = \n",X);
//--- рассчитываем каждое значения вектора X, возведённое в степень InpExponentValue
   X=MathPow(X,InpExponentValue);
   Print("MathPow(X,",(string)InpExponentValue,") = \n",X);
```

© 2000-2025, MetaQuotes Ltd.

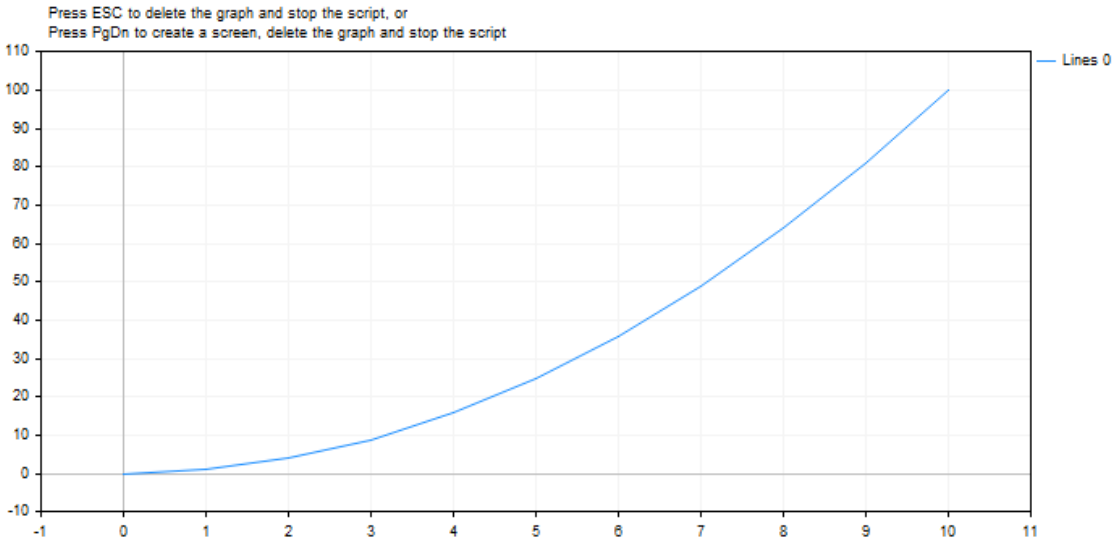
```
//--- переносим рассчитанные значения из вектора в массив
   double y_array[];
   X.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
   /*
   результат:
   vector X =
   [0,1,2,3,4,5,6,7,8,9,10]
   MathPow(X,2.0) =
   [0,1,4,9,16,25,36,49,64,81,100]
   */
  }
//+------------------------------------------------------------------+
//| Заполняет вектор значением value с приращением step              |
//+------------------------------------------------------------------+
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;
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:

© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

MathRand

Возвращает псевдослучайное целое число в диапазоне от 0 до 32767.

```
int  MathRand();
```

Возвращаемое значение

Целое число в диапазоне от 0 до 32767.

Примечание

Перед первым вызовом функции необходимо использовать функцию MathSrand, чтобы перевести генератор псевдослучайных чисел в начальное состояние.

Примечание

Вместо функции MathRand() можно использовать функцию rand().

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- при каждом запуске устанавливаем новое начальное состояние для генерации ряда по
   MathSrand(GetTickCount());
//--- в цикле выводим в журнал 10 сгенерированных псевдослучайных целых чисел
   for(int i=0; i<10; i++)
     {
      int rand_value=MathRand();
      PrintFormat("Псевдослучайное целое число №%d: %u",i+1,rand_value);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

MathRound

Возвращает значение, округленное до ближайшего целого числа указанного числового значения.

```
double  MathRound(
   double  value      // округляемое значение
   );
```

Параметры

value

[in] Числовая величина для округления.

Возвращаемое значение

Значение, округленное до ближайшего целого числа.

Примечание

Вместо функции MathRound() можно использовать функцию round().

Пример:

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявляем переменные для преобразования
   double value=0;                     // вещественное число для MathRound-преобразова
   int    round_value=0;               // сюда получаем результат
//--- в цикле по количеству десятичных приращений вещественного числа
   for(int i=0; i<VALUES_TOTAL; i++)
     {
      //--- увеличиваем значение вещественного числа,
      //--- получаем числовое значение, округленное до ближайшего целого числа
      //--- и выводим в журнал контрольные значения
      value+=0.1;
      round_value=(int)MathRound(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, round value: %d",value,round_value);
      /*
      результат:
      value: 0.1, round value: 0
      value: 0.2, round value: 0
      value: 0.3, round value: 0
      value: 0.4, round value: 0
      value: 0.5, round value: 1
      value: 0.6, round value: 1
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, round value: 1
      value: 0.8, round value: 1
      value: 0.9, round value: 1
      value: 1.0, round value: 1
      value: 1.1, round value: 1
      value: 1.2, round value: 1
      value: 1.3, round value: 1
      value: 1.4, round value: 1
      value: 1.5, round value: 2
      value: 1.6, round value: 2
      value: 1.7, round value: 2
      value: 1.8, round value: 2
      value: 1.9, round value: 2
      value: 2.0, round value: 2
      value: 2.1, round value: 2
      value: 2.2, round value: 2
      value: 2.3, round value: 2
      value: 2.4, round value: 2
      value: 2.5, round value: 3
      value: 2.6, round value: 3
      value: 2.7, round value: 3
      value: 2.8, round value: 3
      value: 2.9, round value: 3
      value: 3.0, round value: 3
      value: 3.1, round value: 3
      */
    }
}
```

© 2000-2025, MetaQuotes Ltd.

MathSin

Возвращает синус указанного угла.

```
double  MathSin(
   double  value      // число
   );
```

Параметры

value

[in] Угол в радианах.

Возвращаемое значение

Синус угла, указанного в радианах. Возвращает значение в диапазоне от -1 до 1.

Примечание

Вместо функции MathSin() можно использовать функцию sin().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- получим 101 значение от 0 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение синуса для каждого значения вектора X
   vector Y=MathSin(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

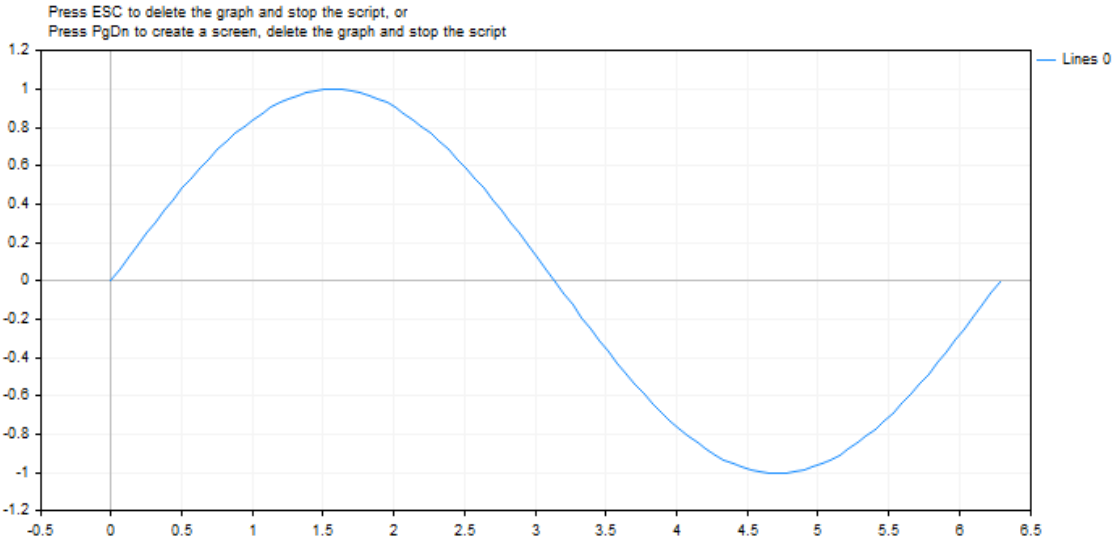
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



© 2000-2025, MetaQuotes Ltd.

MathSqrt

Возвращает квадратный корень числа.

```
double  MathSqrt(
   double  value      // положительное число
   );
```

Параметры

value

[in] Положительная числовая величина.

Возвращаемое значение

Квадратный корень числа value. Если value отрицательно, MathSqrt возвращает NaN (неопределенное значение).

Примечание

Вместо функции MathSqrt() можно использовать функцию sqrt().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим 11 значений от 0 до 10 с шагом 1
   vector X(11,VectorArange);
   Print("vector X = \n",X);
//--- рассчитываем квадратный корень каждого значения вектора X
   X=MathSqrt(X);
   Print("MathSqrt(X) = \n",X);

//--- переносим рассчитанные значения из вектора в массив
   double y_array[];
   X.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

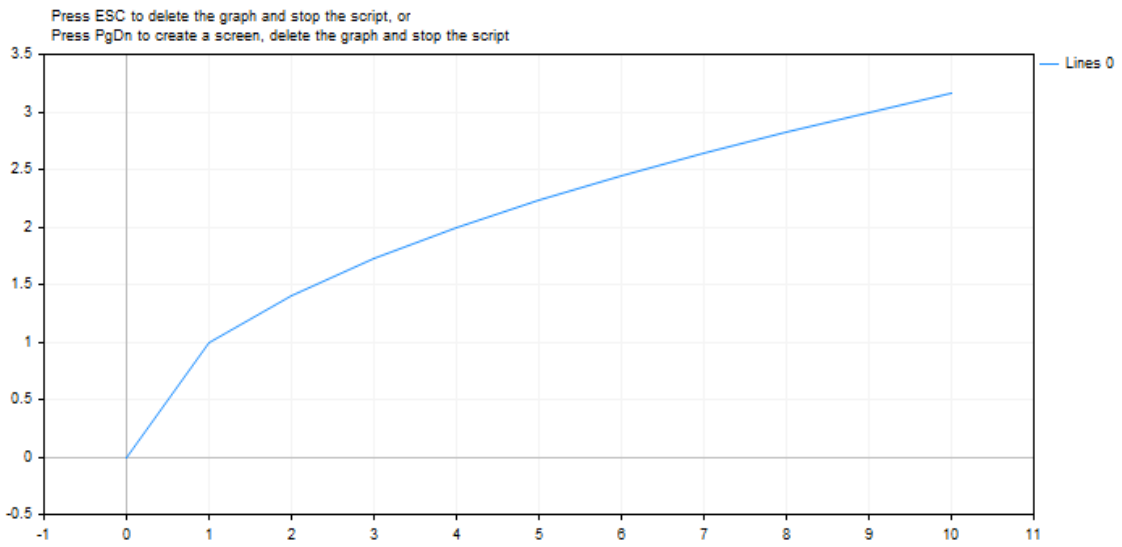
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
   /*
   результат:
   vector X =
   [0,1,2,3,4,5,6,7,8,9,10]
   MathSqrt(X) =
   [0,1,1.414213562373095,1.732050807568877,2,2.23606797749979,2.449489742783178,2.645
   */
  }
//+------------------------------------------------------------------+
//| Заполняет вектор значением value с приращением step              |
//+------------------------------------------------------------------+
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;
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



© 2000-2025, MetaQuotes Ltd.

Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathSrand

Устанавливает начальное состояние для генерации ряда псевдослучайных целых чисел.

```
void  MathSrand(
   int  seed      // инициализирующее число
   );
```

Параметры

seed

[in] Начальное число для ряда случайных чисел.

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Функция MathRand() предназначена для генерации последовательности псевдослучайных чисел. Вызов MathSrand() с определенным инициализирующим числом позволяет получать всегда одну и ту же последовательность псевдослучайных чисел.

Для гарантированного получения неповторяющейся последовательности используйте вызов MathSrand(GetTickCount()), так как значение GetTickCount() увеличивается с момента запуска операционной системы и не повторяется в течение 49 дней, пока не переполнится встроенный счетчик миллисекунд. Использование MathSrand(TimeCurrent()) не подходит по той причине, что функция TimeCurrent() возвращает время прихода последнего тика, которое может не меняться долгое время, например, в выходные дни.

Инициализацию генератора псевдослучайных чисел с помощью MathSrand() для индикаторов и экспертов лучше всего делать в обработчике OnInit(), это избавит от последующих многократных перезапусков генератора в OnTick() и OnCalculate().

Вместо функции MathSrand() можно использовать функцию srand().

Пример:

```
#property description "Индикатор демонстрирует центральную предельную  теорему, котора
#property description "Сумма достаточно большого количества слабо зависимых случайных
#property description "имеющих примерно одинаковые масштабы (ни одно из слагаемых не д
#property description "не вносит в сумму определяющего вклада), имеет распределение, б

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- свойства графического построения
#property indicator_label1  "Label"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRoyalBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  5
```

© 2000-2025, MetaQuotes Ltd.

```
//--- input переменная
input int      sample_number=10;
//--- индикаторный буфер для отрисовки распределения
double         LabelBuffer[];
//--- счетчик тиков
double         ticks_counter;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- связывание массива и индикаторного буфера
   SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- развернем индикаторный буфер из настоящего в прошлое
   ArraySetAsSeries(LabelBuffer,true);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- инициализируем счетчик тиков
   ticks_counter=0;
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- при нулевом счетчике обнулим буфер индикатора
   if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- увеличим счетчик
   ticks_counter++;
//--- нужно периодически сбрасывать счетчик тиков, чтобы оживлять распределение
   if(ticks_counter>100)
     {
      Print("Обнулили значения индикатора, начнем заполнять ячейки снова");
      ticks_counter=0;
     }
//--- получим выборку случайных значений, как сумму трех чисел от 0 до 7
   for(int i=0;i<sample_number;i++)
     {
      //--- вычисление индекса ячейки, куда выпадет случайное число как сумма трех дру
      int rand_index=0;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- получим три случайных числа от 0 до 7
      for(int k=0;k<3;k++)
        {
         //--- остаток от деления на 7 вернет значение от 0 до 6
         rand_index+=MathRand()%7;
        }
      //--- увеличим на единицу значение в ячейке с номером rand_index
      LabelBuffer[rand_index]++;
     }
//--- выход из обработчика OnCalculate()
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathTan

Возвращает тангенс числа.

```
double  MathTan(
   double  rad      // аргумент в радианах
   );
```

Параметры

rad

[in] Угол в радианах.

Возвращаемое значение

Тангенс числа rad. Если rad больше или равен 263 или меньше или равен -263, то происходит потеря значения и функция возвращает неопределенное число.

Примечание

Вместо функции MathTan() можно использовать функцию tan().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=-263;
//--- получим 101 значение от -263 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение тангенса для каждого значения вектора X
   vector Y=MathTan(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

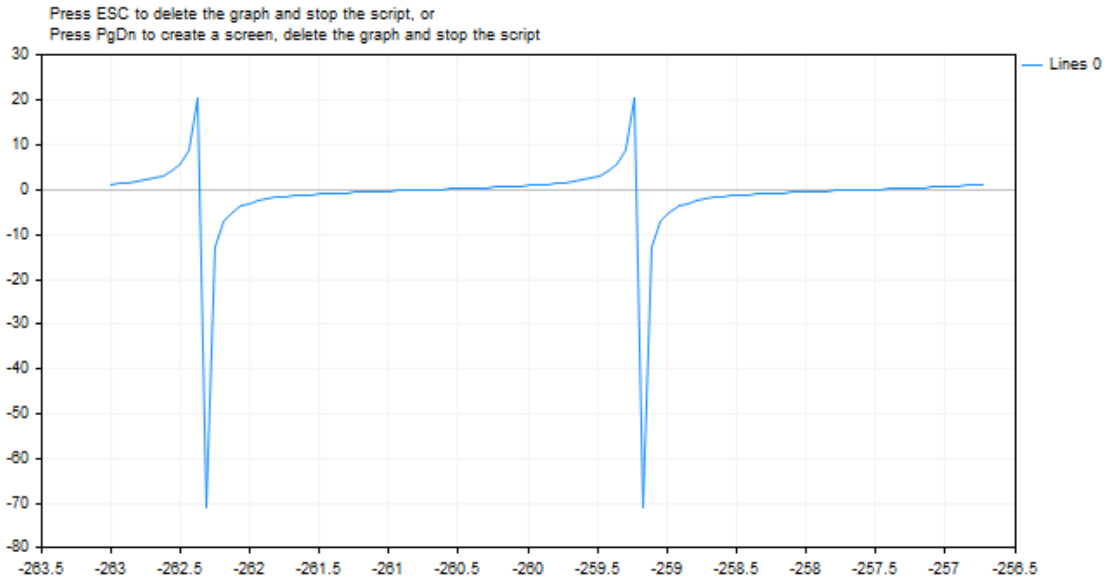
© 2000-2025, MetaQuotes Ltd.

```
//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathIsValidNumber

Проверяет корректность действительного числа

```
bool  MathIsValidNumber(
   double  number      // число для проверки
   );
```

Параметры

number

[in] Проверяемое число.

Возвращаемое значение

Возвращает true, если проверяемое значение является допустимым вещественным числом. Если проверяемое значение является плюс или минус бесконечностью, либо "не числом" (NaN - not a number), функция возвращает false.

Пример:

```
   double abnormal=MathArcsin(2.0);
   if(!MathIsValidNumber(abnormal)) Print("Внимание! MathArcsin(2.0) = ",abnormal);
```

Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathExpm1

Возвращает значение выражения MathExp(x)-1.

```
double  MathExpm1(
   double  value      // степень для числа e
   );
```

Параметры

value

[in] Число, определяющее степень.

Возвращаемое значение

Число типа double. При переполнении функция возвращает INF (бесконечность), в случае потери порядка точности MathExpm1 возвращает 0.

Примечание

При значениях x близком к 0, функция MathExpm1(x) даёт гораздо более точные значения, чем MathExp(x)-1.

Вместо функции MathExpm1() можно использовать функцию expm1().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим 9 значений от 0 до 8 с шагом 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- рассчитываем значение ("e"(число Эйлера) в степени каждого значения вектора X)
   X=MathExpm1(X);
   Print("MathExpm1(X) = \n",X);

//--- переносим рассчитанные значения из вектора в массив
   double y_array[];
   X.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
```

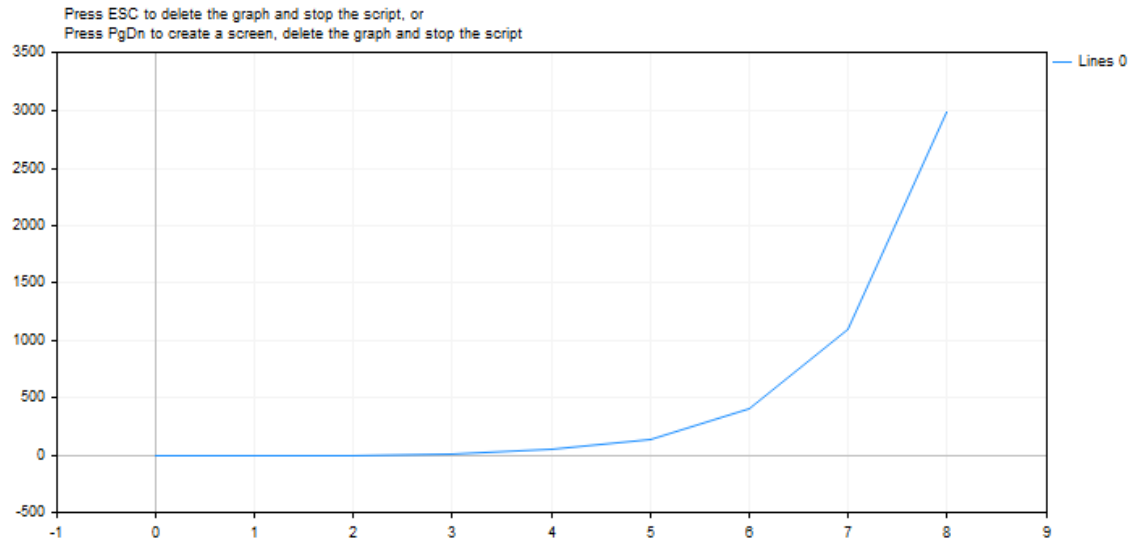
© 2000-2025, MetaQuotes Ltd.

```
   CurvePlot(y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| Заполняет вектор значением value с приращением step              |
//+------------------------------------------------------------------+
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;
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathLog1p

Возвращает значение выражения MathLog(1+x).

```
double  MathLog1p(
   double  value      // число для взятия логарифма
   );
```

Параметры

val

[in] Значение, логарифм которого должен быть вычислен.

Возвращаемое значение

Натуральный логарифм значения (value+1) в случае успеха. Если value < -1, то функция возвращает NaN (неопределенное значение). Если value равно -1, функция возвращает INF (бесконечность) .

Примечание

При значениях x близком к 0, функция MathLog1p(x) даёт гораздо более точные значения, чем MathLog(1+x).

Вместо функции MathLog1p() можно использовать функцию log1p().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим 9 значений от 0 до 8 с шагом 1
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- для каждого значения вектора рассчитываем логарифм для выражения (1 + значение в
   X=MathLog1p(X);
   Print("MathLog1p(X) = \n",X);

//--- переносим рассчитанные значения из вектора в массив
   double y_array[];
   X.Swap(y_array);
```

© 2000-2025, MetaQuotes Ltd.

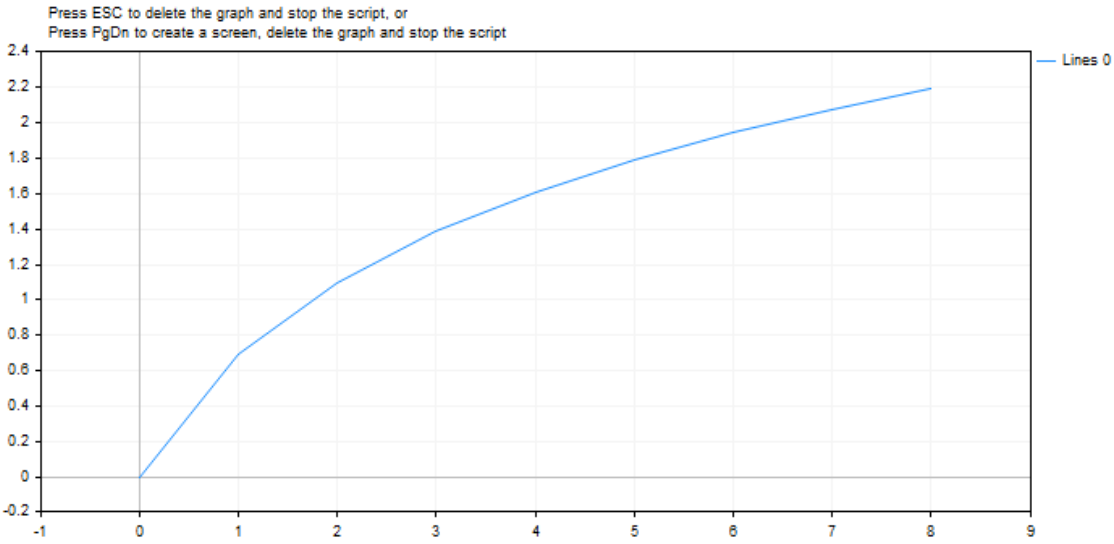
```
//--- нарисуем график рассчитанных значений вектора
   CurvePlot(y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
   /*
   результат:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog1p(X) =
   [0,0.6931471805599453,1.09861228866811,1.386294361119891,1.6094379124341,1.7917594
   */
  }
//+------------------------------------------------------------------+
//| Заполняет вектор значением value с приращением step              |
//+------------------------------------------------------------------+
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;
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:

© 2000-2025, MetaQuotes Ltd.



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArccosh

Возвращает значение гиперболического арккосинуса.

```
double  MathArccosh(
   double  value     // 1 <= value < ∞
   );
```

Параметры

val

[in] Значение value, гиперболический арккосинус которого должен быть вычислен.

Возвращаемое значение

Гиперболический арккосинус числа. Если value меньше +1, функция возвращает NaN (неопределенное значение).

Примечание

Вместо функции MathArccosh() можно использовать функцию acosh().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/10);
   delta[0]=1;
//--- получим 101 значение от 1 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение гиперболического арккосинуса для каждого значения вектора
   vector Y=MathArccosh(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

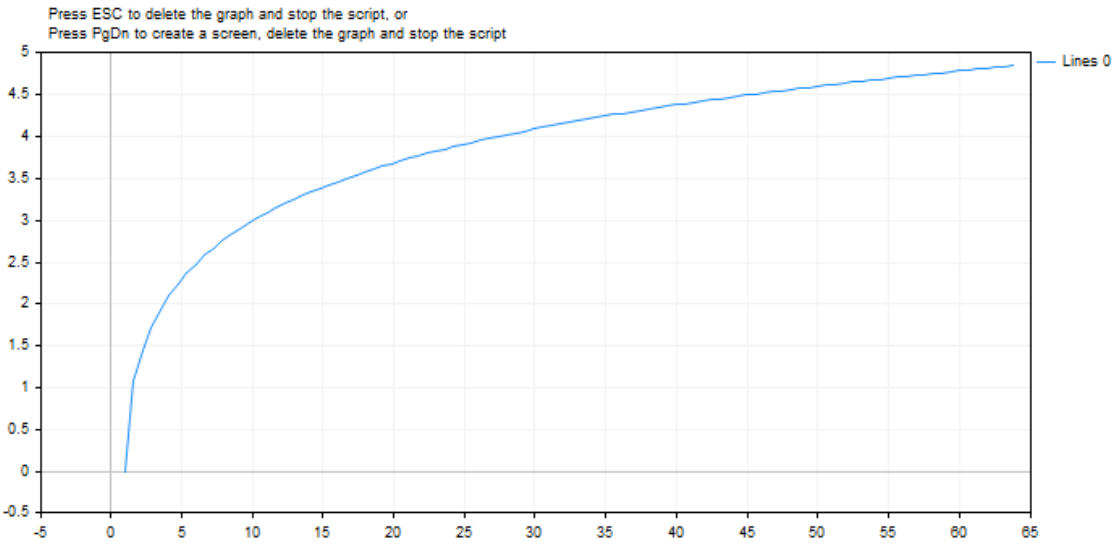
© 2000-2025, MetaQuotes Ltd.

```
//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArcsinh

Возвращает значение гиперболического арксинуса.

```
double  MathArcsinh(
   double  value      // -∞ < value < +∞
   );
```

Параметры

val

[in] Значение value, арксинус которого должен быть вычислен.

Возвращаемое значение

Гиперболический арксинус числа.

Примечание

Вместо функции MathArcsinh() можно использовать функцию asinh().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/10);
   delta[0]=-31;
//--- получим 101 значение от -31 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение гиперболического арксинуса для каждого значения вектора X
   vector Y=MathArcsinh(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

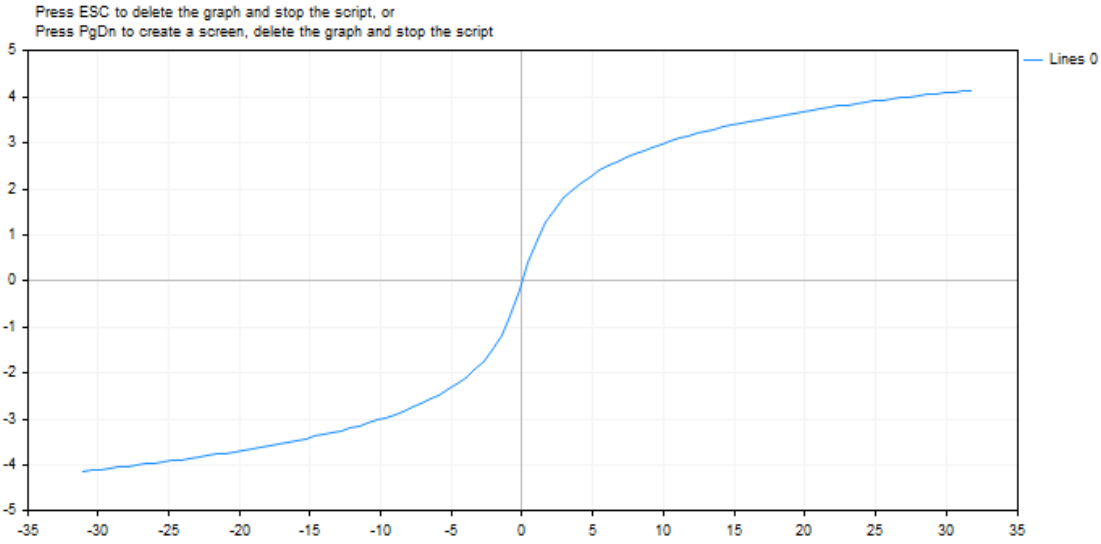
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArctanh

Возвращает значение гиперболического арктангенса.

```
double  MathArctanh(
   double  value      // значение в дипазоне -1 < value < 1
   );
```

Параметры

value

[in] Число в дипазоне -1 < value < 1, представляющее тангенс.

Возвращаемое значение

Гиперболический арктангенс числа.

Примечание

Вместо функции MathArctanh() можно использовать функцию atanh().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(34,2*M_PI/100);
   delta[0]=-1;
//--- получим 34 значения от -1 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение гиперболического арктангенса для каждого значения вектора
   vector Y=MathArctanh(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

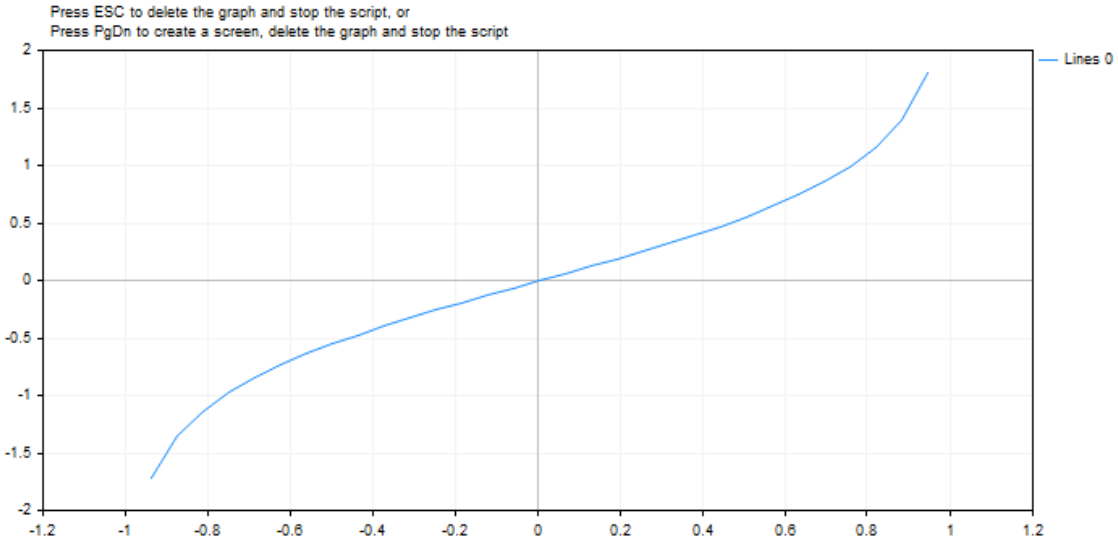
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



© 2000-2025, MetaQuotes Ltd.

MathCosh

Возвращает гиперболический косинус числа.

```
double  MathCosh(
   double  value      // число
   );
```

Параметры

value

[in] Значение.

Возвращаемое значение

Гиперболический косинус числа, значение в диапазоне от +1 до плюс бесконечности.

Примечание

Вместо функции MathCosh() можно использовать функцию cosh().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- получим 101 значение от 0 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение гиперболического косинуса для каждого значения вектора X
   vector Y=MathCosh(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

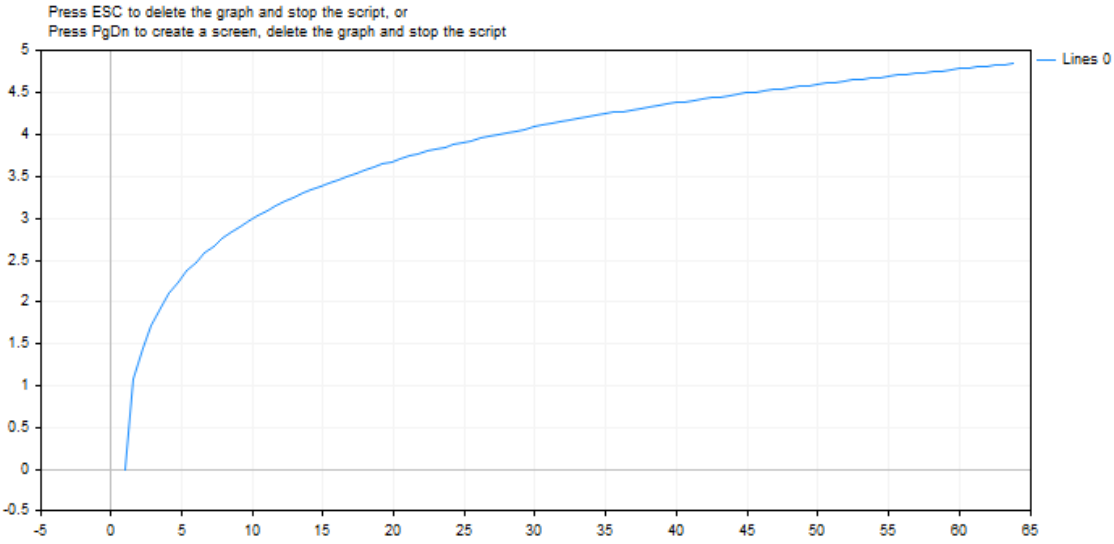
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



© 2000-2025, MetaQuotes Ltd.

MathSinh

Возвращает гиперболический синус числа.

```
double  MathSinh(
   double  value      // число
   );
```

Параметры

value

[in] Значение.

Возвращаемое значение

Гиперболический синус числа.

Примечание

Вместо функции MathSinh() можно использовать функцию sinh().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- получим 101 значение от 0 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение гиперболического синуса для каждого значения вектора X
   vector Y=MathSinh(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

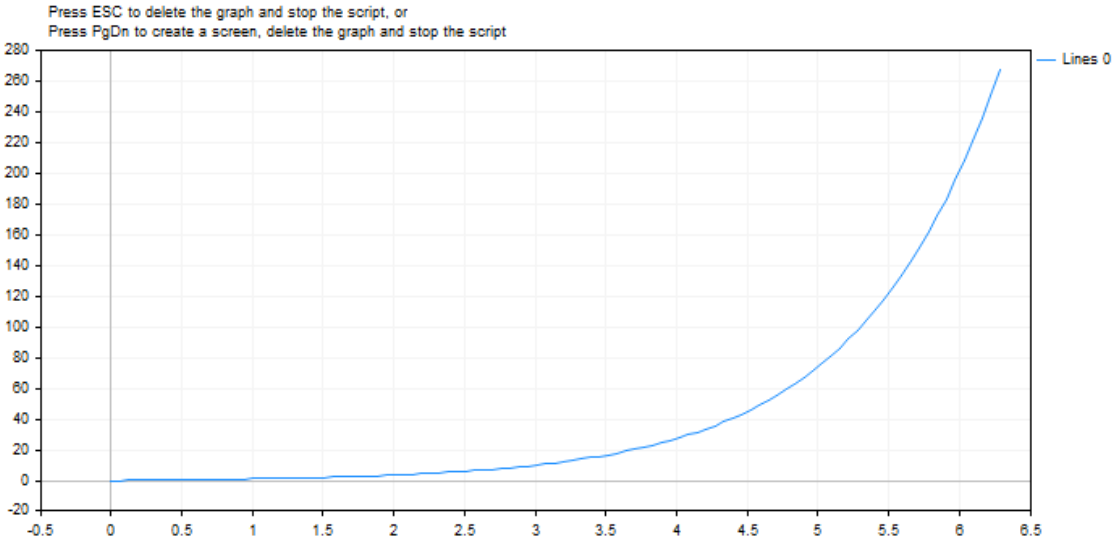
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



© 2000-2025, MetaQuotes Ltd.

MathTanh

Возвращает гиперболический тангенс числа.

```
double  MathTanh(
   double  value      // число
   );
```

Параметры

value

[in] Значение.

Возвращаемое значение

Гиперболический тангенс числа, значение в диапазоне от -1 до +1.

Примечание

Вместо функции MathTanh() можно использовать функцию tanh().

Пример:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- получим 101 значение от 0 до 2 пи с шагом delta
   vector X=delta.CumSum();
//--- рассчитываем значение гиперболического тангенса для каждого значения вектора X
   vector Y=MathTanh(X);

//--- переносим рассчитанные значения из векторов в массивы
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- нарисуем график рассчитанных значений вектора
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- ожидаем нажатия клавиш Escape или PgDn для удаления графика (создания скриншота)
```

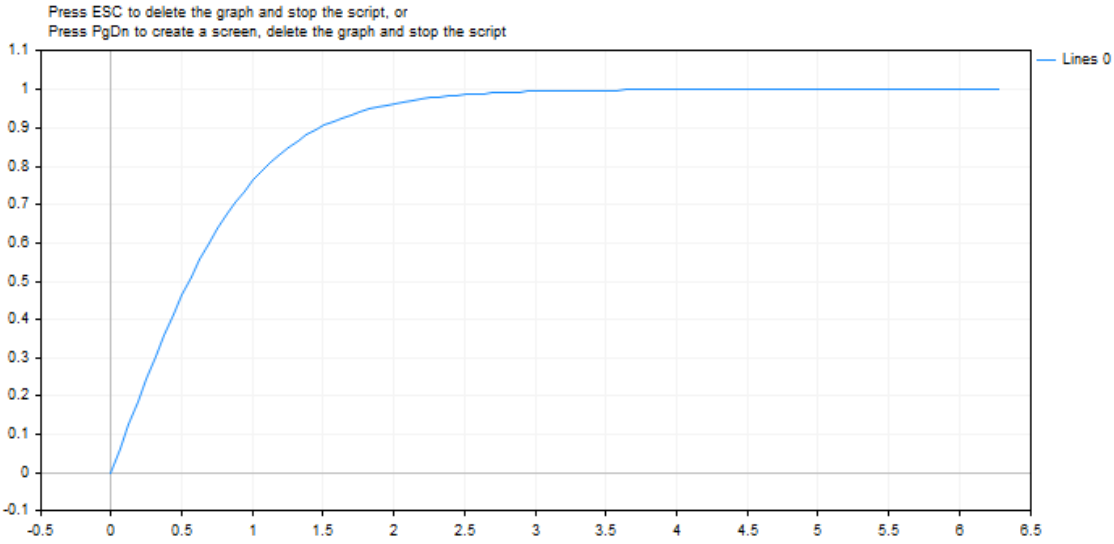
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- чистим за собой
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| При нажатии ESC возвращает true                                  |
//| При нажатии PgDn делает скриншот графика и возвращает true       |
//| Иначе - возвращает false                                         |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- если нажата ESC - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- если нажата PgDn и успешно сделан скриншот графика - возвращаем true
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- возвращаем false
   return(false);
  }
//+------------------------------------------------------------------+
//| Создаёт объект-график и рисует кривую                            |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
   string text1="Press ESC to delete the graph and stop the script, or";
   string text2="Press PgDn to create a screen, delete the graph and stop the script";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| Создаёт скриншот экрана и сохраняет изображение в файл           |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

Результат:



Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

MathSwap

Меняет порядок байтов в значении типа ushort.

```
ushort  MathSwap(
   ushort  value      // значение
   );
```

Параметры

value

[in] Значение для смены порядка байтов.

Возвращаемое значение

Значение ushort с обратным порядком байтов.

MathSwap

Меняет порядок байтов в значении типа uint.

```
uint  MathSwap(
   uint   value      // значение
   );
```

Параметры

value

[in] Значение для смены порядка байтов.

Возвращаемое значение

Значение uint с обратным порядком байтов.

MathSwap

Меняет порядок байтов в значении типа ulong.

```
ulong  MathSwap(
   ulong  value      // значение
   );
```

Параметры

value

[in] Значение для смены порядка байтов.

Возвращаемое значение

Значение ulong с обратным порядком байтов.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
#property script_show_inputs

input ulong  InpLongValue  =  1;     // Enter any ulong value here
input uint   InpIntValue   =  2;     // Enter any uint value here
input ushort InpShortValue =  3;     // Enter any ushort value here

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

//--- распечатаем в журнале введённые и преобразованные MathSwap() значения в десятичн
   Print(ValueDescription(InpLongValue));
   Print(ValueDescription(InpIntValue));
   Print(ValueDescription(InpShortValue));
   /*
   результат:
   ulong value: 1
   ulong value: 72057594037927936 using MathSwap()
   binary ulong value: 0000000000000000000000000000000000000000000000000000000000000000
   binary ulong value: 0000000100000000000000000000000000000000000000000000000000000000

   uint value: 2
   uint value: 33554432 using MathSwap()
   binary uint value: 00000000000000000000000000000010
   binary uint value: 00000010000000000000000000000000 using MathSwap()

   ushort value: 3
   ushort value: 768 using MathSwap()
   binary ushort value: 0000000000000011
   binary ushort value: 0000001100000000 using MathSwap()
   */
  }
//+------------------------------------------------------------------+
//| Возвращает текст описания значений переменной                    |
//+------------------------------------------------------------------+
template <typename T>
string ValueDescription(T x)
  {
   int    num_bits   = sizeof(T)*8;
   string type_name  = typename(T);
   string bin_x      = NumberToBinaryString(x);
   string bin_swap_x = NumberToBinaryString(MathSwap(x));
   return(StringFormat("%s value: %lld\n%s value: %lld using MathSwap()\nbinary %s va
          type_name, x, type_name, MathSwap(x), type_name, num_bits, bin_x, type_name,
  }
//+------------------------------------------------------------------+
//| Возвращает бинарное представление числа в виде строки            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
template <typename T>
string NumberToBinaryString(T x)
  {
   string res  = "";
   int    i    = -1;
   uchar  size = sizeof(T)*8-1;
   ulong  mask = (ulong)1<<size;
   while(!((x<<++i)& mask));
   for(; i <=size; i++)
      res += !((x<<i)& mask) ? "0" : "1";
   return res;
  }
```

Смотри также

Сетевые функции, SocketRead, SocketSend, SocketTlsRead, SocketTlsReadAvailable, SocketTlsSend

© 2000-2025, MetaQuotes Ltd.

Строковые функции

Группа функций, предназначенных для работы с данными типа string.

Функция	Действие
StringAdd	Присоединяет к концу строки по месту указанную подстроку
StringBufferLen	Возвращает размер буфера, распределенного для строки
StringCompare	Сравнивает две строки и возвращает 1, если первая строка больше второй; 0 - если строки равны; -1 (минус один) - если первая строка меньше второй
StringConcatenate	Формирует строку из переданных параметров
StringFill	Заполняет указанную строку по месту указанными символами
StringFind	Поиск подстроки в строке
StringGetCharacter	Возвращает значение символа, расположенного в указанной позиции строки
StringInit	Инициализирует строку указанными символами и обеспечивает указанный размер строки
StringLen	Возвращает число символов в строке
StringSetLength	Устанавливает для строки указанную длину в символах
StringReplace	Заменяет в строке все найденные подстроки на заданную последовательность символов.
StringReserve	Резервирует в памяти для строки буфер указанного размера
StringSetCharacter	Возвращает копию строки с измененным значением символа в указанной позиции
StringSplit	Получает из указанной строки подстроки по заданному разделителю и возвращает количество полученных подстрок
StringSubstr	Извлекает подстроку из текстовой строки, начинающейся с указанной позиции
StringToLower	Преобразует все символы указанной строки в строчные (маленькие) по месту
StringToUpper	Преобразует все символы указанной строки в прописные (большие) по месту
StringTrimLeft	Удаляет символы перевода каретки, пробелы и символы табуляции с начала строки
StringTrimRight	Удаляет символы перевода каретки, пробелы и символы табуляции в конце строки

© 2000-2025, MetaQuotes Ltd.

StringAdd

Присоединяет к концу строки по месту указанную подстроку.

```
bool  StringAdd(
   string&  string_var,         // строка, к которой добавляем
   string   add_substring       // добавляемая строка
   );
```

Параметры

string_var

[in][out] Строка, которая будет дополнена.

add_substring

[in] Строка, которая будет добавлена в конец исходной строки.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Пример:

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

//--- второй способ
   start=GetTickCount();
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   stop=GetTickCount();
   Print("time for 'StringAdd(a,b)' = ",(stop-start)," milliseconds, i = ",i);

//--- третий способ
   start=GetTickCount();
   a="a"; // заново инициализируем переменную a
   for(i=0;i<length;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       StringConcatenate(c,a,b);
      }
   stop=GetTickCount();
   Print("time for 'StringConcatenate(c,a,b)' = ",(stop-start)," milliseconds, i = ",i
  }
```

Смотри также

StringConcatenate, StringSplit, StringSubstr

© 2000-2025, MetaQuotes Ltd.

StringBufferLen

Возвращает размер буфера, распределенного для строки.

```
int  StringBufferLen(
   string  string_var      // строка
   )
```

Параметры

string_var

[in] Строка.

Возвращаемое значение

Значение 0 означает, что строка - константная и содержимое буфера менять нельзя. -1 означает, что строка принадлежит клиентскому терминалу и изменение содержимого буфера может привести к неопределенным результатам.

Пример:

```
void OnStart()
  {
   long length=1000;
   string a="a",b="b";
//---
   long i;
   Print("before: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   Print("after: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = +-",StringLen(a));
  }
```

Смотри также

StringAdd, StringInit, StringLen, StringFill

© 2000-2025, MetaQuotes Ltd.

StringCompare

Сравнивает между собой две строки и возвращает результат сравнения в виде целого числа.

```
int  StringCompare(
   const string&  string1,                // первая сравниваемая строка
   const string&  string2,                // вторая сравниваемая строка
   bool           case_sensitive=true     // режим учета регистра букв при сравнении
   );
```

Параметры

string1

[in] Первая строка.

string2

[in] Вторая строка.

case_sensitive=true

[in] Режим учета регистра букв. Если значения равно true, то "A">"a". Если значение равно false, то "A"="a". По умолчанию значение параметра равно true.

Возвращаемое значение

- -1 (минус один), если string1<string2
- 0 (ноль), если string1=string2
- 1 (один), если string1>string2

Примечание

Строки сравниваются посимвольно, символы сравниваются в алфавитном порядке в соответствии с текущей кодовой страницей.

Пример:

```
void OnStart()
  {
//--- что больше, яблоко или дом?
   string s1="Apple";
   string s2="home";

//--- сравним с учетом регистра
   int result1=StringCompare(s1,s2);
   if(result1>0) PrintFormat("Сравнение с учетом регистра: %s > %s",s1,s2);
   else
     {
      if(result1<0)PrintFormat("Сравнение с учетом регистра: %s < %s",s1,s2);
      else PrintFormat("Сравнение с учетом регистра: %s = %s",s1,s2);
     }

//--- сравним без учета регистра
   int result2=StringCompare(s1,s2,false);
   if(result2>0) PrintFormat("Сравнение без учета регистра: %s > %s",s1,s2);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       if(result2<0)PrintFormat("Сравнение без учета регистра: %s < %s",s1,s2);
       else PrintFormat("Сравнение без учета регистра: %s = %s",s1,s2);
      }
/* Результат
     Сравнение с учетом регистра: Apple < home
     Сравнение без учета регистра: Apple < home
*/
  }
```

Смотри также

Тип string, CharToString(), ShortToString(), StringToCharArray(), StringToShortArray(), StringGetCharacter(), Использование кодовой страницы

© 2000-2025, MetaQuotes Ltd.

StringConcatenate

Формирует строку из переданных параметров и возвращает размер сформированной строки. Параметры могут иметь любой тип. Количество параметров не может быть меньше 2 и не может превышать 64.

```
int  StringConcatenate(
   string&  string_var,    // строка для формирования
   void argument1          // первый параметр любого простого типа
   void argument2          // второй параметр любого простого типа
   ...                     // следующий параметр любого простого типа
   );
```

Параметры

string_var

[out] Строка, которая будет сформирована в результате конкатенации.

argumentN

[in] Любые значения, разделенные запятыми. От 2 до 63 параметров любого простого типа.

Возвращаемое значение

Возвращает длину строки, сформированной путем конкатенации преобразованных в тип string параметров. Параметры преобразуются в строки по тем же правилам, что и в функциях Print() и Comment().

Пример:

```
void OnStart()
  {
//--- объявляем и определяем переменные, участвующие в конкатенации
   string text="";
   string text1="This script shows how the StringConcatenate() function works.\n";
   string text2="This is the second line, at the end of which there is a line break co
   string text3="This is line number ";
   int    num3=3;
   string text31=", the number of which is entered into the function as a separate par
   string textN="\n";
   string text4="This is line number 4, preceded by a separate parameter with a line b
   int    length=StringConcatenate(text, text1, text2, text3, num3, text31, textN, tex
   Print(text, "\nLength of the resulting string = ", length);

   /*
   Результат
   This script shows how the StringConcatenate() function works.
   This is the second line, at the end of which there is a line break control code.
   This is line number 3, the number of which is entered into the function as a separa
   This is line number 4, preceded by a separate parameter with a line break code.
   Line 5 includes a real number: 0.12345
   Length of the resulting string = 358
```

© 2000-2025, MetaQuotes Ltd.

```
   */
  }
```

Смотри также

StringAdd, StringSplit, StringSubstr

© 2000-2025, MetaQuotes Ltd.

StringFill

Заполняет указанную строку по месту указанными символами.

```
bool  StringFill(
   string&   string_var,       // строка для заполнения
   ushort    character         // символ, которым будет заполнена строка
   );
```

Параметры

string_var

[in][out] Строка, которая будет заполнена указанным символом.

character

[in] Символ, которым будет заполнена строка.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Заполнение строки по месту означает, что символы вставляются непосредственно в строку без промежуточных операций создания новой строки и копирования. Это позволяет уменьшить время работы со строкой в данной функции.

Пример:

```
void OnStart()
  {
   string str;
   StringInit(str,20,'_');
   Print("str = ",str);
   StringFill(str,0);
   Print("str = ",str,": StringBufferLen(str) = ", StringBufferLen(str));
  }
// Результат
//   str = ____________________
//   str =  : StringBufferLen(str) = 20
//
```

Смотри также

StringBufferLen, StringLen, StringInit

© 2000-2025, MetaQuotes Ltd.

StringFind

Поиск подстроки в строке.

```
int  StringFind(
   string  string_value,        // строка, в которой ищем
   string  match_substring,     // что ищем
   int     start_pos=0          // с какой позиции начинать поиск
   );
```

Параметры

string_value

[in] Строка, в которой производится поиск.

match_substring

[in] Искомая подстрока.

start_pos=0

[in] Позиция в строке, с которой должен быть начат поиск.

Возвращаемое значение

Возвращает номер позиции в строке, с которой начинается искомая подстрока, либо -1, если подстрока не найдена.

Пример:

```
#define   RESERVE    100

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим базовую валюту инструмента и валюту прибыли
   string symbol_currency_base  =SymbolInfoString(Symbol(), SYMBOL_CURRENCY_BASE);
   string symbol_currency_profit=SymbolInfoString(Symbol(), SYMBOL_CURRENCY_PROFIT);
   PrintFormat("Symbol Currency Base: %s\nSymbol Currency Profit: %s", symbol_currency

//--- в цикле по всем имеющимся на сервере символам
   int total=SymbolsTotal(false), pos=-1;
   for(int i=0; i<total; i++)
     {
      //--- получаем имя очередного символа
      string name=SymbolName(i, false);

      //--- ищем в имени символа подстроку с наименованием базовой валюты и
      //--- если подстрока найдена - выведем в журнал наименование инструмента, его ин
      pos = StringFind(name, symbol_currency_base);
      if(pos >= 0)
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("The '%s' symbol at index %u in the list contains the '%s' currer

      //--- ищем в имени символа подстроку с наименованием котируемой валюты и
      //--- если подстрока найдена - выведем в журнал наименование инструмента, его ин
      pos = StringFind(name, symbol_currency_profit);
      if(pos >= 0)
         PrintFormat("The '%s' symbol at index %u in the list contains the '%s' currer
   }

/*
Результат
StringFind (EURUSD,D1)   Symbol Currency Base: EUR
StringFind (EURUSD,D1)   Symbol Currency Profit: USD
The 'EURUSD' symbol at index 0 in the list contains the 'EUR' currency. Substring p
The 'EURUSD' symbol at index 0 in the list contains the 'USD' currency. Substring p
The 'GBPUSD' symbol at index 1 in the list contains the 'USD' currency. Substring p
The 'USDCHF' symbol at index 2 in the list contains the 'USD' currency. Substring p
The 'USDJPY' symbol at index 3 in the list contains the 'USD' currency. Substring p
The 'USDCNH' symbol at index 4 in the list contains the 'USD' currency. Substring p
The 'USDRUB' symbol at index 5 in the list contains the 'USD' currency. Substring p
The 'AUDUSD' symbol at index 6 in the list contains the 'USD' currency. Substring p
The 'NZDUSD' symbol at index 7 in the list contains the 'USD' currency. Substring p
The 'USDCAD' symbol at index 8 in the list contains the 'USD' currency. Substring p
The 'USDSEK' symbol at index 9 in the list contains the 'USD' currency. Substring p
The 'USDHKD' symbol at index 10 in the list contains the 'USD' currency. Substring
The 'USDSGD' symbol at index 11 in the list contains the 'USD' currency. Substring
The 'USDNOK' symbol at index 12 in the list contains the 'USD' currency. Substring
The 'USDDKK' symbol at index 13 in the list contains the 'USD' currency. Substring
The 'USDTRY' symbol at index 14 in the list contains the 'USD' currency. Substring
The 'USDZAR' symbol at index 15 in the list contains the 'USD' currency. Substring
The 'USDCZK' symbol at index 16 in the list contains the 'USD' currency. Substring
The 'USDHUF' symbol at index 17 in the list contains the 'USD' currency. Substring
The 'USDPLN' symbol at index 18 in the list contains the 'USD' currency. Substring
The 'USDRUR' symbol at index 19 in the list contains the 'USD' currency. Substring
The 'EURAUD' symbol at index 27 in the list contains the 'EUR' currency. Substring
The 'EURCAD' symbol at index 28 in the list contains the 'EUR' currency. Substring
The 'EURCHF' symbol at index 29 in the list contains the 'EUR' currency. Substring
The 'EURCZK' symbol at index 30 in the list contains the 'EUR' currency. Substring
The 'EURDKK' symbol at index 31 in the list contains the 'EUR' currency. Substring
The 'EURGBP' symbol at index 32 in the list contains the 'EUR' currency. Substring
The 'EURHKD' symbol at index 33 in the list contains the 'EUR' currency. Substring
The 'EURHUF' symbol at index 34 in the list contains the 'EUR' currency. Substring
The 'EURJPY' symbol at index 35 in the list contains the 'EUR' currency. Substring
The 'EURNOK' symbol at index 36 in the list contains the 'EUR' currency. Substring
The 'EURNZD' symbol at index 37 in the list contains the 'EUR' currency. Substring
The 'EURPLN' symbol at index 38 in the list contains the 'EUR' currency. Substring
The 'EURRUR' symbol at index 39 in the list contains the 'EUR' currency. Substring
The 'EURRUB' symbol at index 40 in the list contains the 'EUR' currency. Substring
The 'EURSEK' symbol at index 41 in the list contains the 'EUR' currency. Substring
```

© 2000-2025, MetaQuotes Ltd.

```
   The 'EURTRY' symbol at index 42 in the list contains the 'EUR' currency. Substring
   The 'EURZAR' symbol at index 43 in the list contains the 'EUR' currency. Substring
   The 'XAUUSD' symbol at index 47 in the list contains the 'USD' currency. Substring
   The 'XAUEUR' symbol at index 48 in the list contains the 'EUR' currency. Substring
   The 'XAGUSD' symbol at index 50 in the list contains the 'USD' currency. Substring
   The 'XAGEUR' symbol at index 51 in the list contains the 'EUR' currency. Substring
   The 'USDCRE' symbol at index 53 in the list contains the 'USD' currency. Substring
   The 'XPDUSD' symbol at index 65 in the list contains the 'USD' currency. Substring
   The 'XPTUSD' symbol at index 66 in the list contains the 'USD' currency. Substring
   The 'USDGEL' symbol at index 67 in the list contains the 'USD' currency. Substring
   The 'USDMXN' symbol at index 68 in the list contains the 'USD' currency. Substring
   The 'EURMXN' symbol at index 69 in the list contains the 'EUR' currency. Substring
   The 'USDCOP' symbol at index 75 in the list contains the 'USD' currency. Substring
   The 'USDARS' symbol at index 76 in the list contains the 'USD' currency. Substring
   The 'USDCLP' symbol at index 77 in the list contains the 'USD' currency. Substring
   The 'EURSGD' symbol at index 89 in the list contains the 'EUR' currency. Substring
   The 'USDILS' symbol at index 95 in the list contains the 'USD' currency. Substring
   The 'USDTHB' symbol at index 122 in the list contains the 'USD' currency. Substring
   The 'USDRMB' symbol at index 123 in the list contains the 'USD' currency. Substring
   The 'EURILS' symbol at index 126 in the list contains the 'EUR' currency. Substring
   The 'EURCNH' symbol at index 137 in the list contains the 'EUR' currency. Substring
   The 'USDBRL' symbol at index 139 in the list contains the 'USD' currency. Substring
   */
  }
```

Смотри также

StringSubstr, StringGetCharacter, StringLen, StringLen

© 2000-2025, MetaQuotes Ltd.

StringGetCharacter

Возвращает значение символа, расположенного в указанной позиции строки.

```
ushort  StringGetCharacter(
   string  string_value,     // строка
   int     pos               // позиция символа в строке
   );
```

Параметры

string_value

[in] Строка.

pos

[in] Позиция символа в строке. Может быть от 0 до StringLen(text) -1.

Возвращаемое значение

Код символа либо 0 в случае какой-либо ошибки. Для получения кода ошибки нужно вызвать функцию GetLastError().

Пример:

```
void OnStart()
  {
//--- сотрём все комментарии на графике
   Comment("");
//--- объявляем строку, из которой будем получать значения кодов символов и запоминаем
   string message = "The script demonstrates the operation of the StringGetCharacter()
   int    length  = StringLen(message);
//--- объявляем строковую переменную, в которую будем добавлять полученные символы из
   string text    = "";
//--- в цикле по размеру демонстрационной строки
   for(int i=0; i<length; i++)
     {
      //--- ждём 1/10 секунды
      Sleep(100);
      //--- получаем символ из строки, расположенный по индексу цикла в демонстрационн
      ushort char_code=StringGetCharacter(message, i);
      //--- добавляем символ в выводимую строку и выводим полученную строку в коммента
      text+=ShortToString(char_code);
      Comment(text);
     }
//--- подождём две секунды и удалим комментарий с графика
   Sleep(2000);
   Comment("");

   /*
   Результат: побуквенное появление на экране демонстрационной строки
   The script demonstrates the operation of the StringGetCharacter() function
```

© 2000-2025, MetaQuotes Ltd.

```
   */
  }
```

Смотри также

StringSetCharacter, StringBufferLen, StringLen, StringFill, StringInit, StringToCharArray, StringToShortArray

© 2000-2025, MetaQuotes Ltd.

StringInit

Инициализирует строку указанными символами и обеспечивает указанный размер строки.

```
bool  StringInit(
   string&    string_var,       // строка для инициализации
   int        new_len=0,        // требуемая длина строки после инициализации
   ushort     character=0       // символ, которым будет заполнена строка
   );
```

Параметры

string_var

[in][out] Строка, которая должна быть инициализирована или деинициализирована.

new_len=0

[in] Длина строки после инициализации. Если размер=0, то деинициализирует строку, то есть, буфер строки освобождается и адрес буфера обнуляется.

character=0

[in] Символ для заполнения строки.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Если character=0 и размер new_len>0, то будет распределен буфер строки указанного размера и заполнен нулями. Размер строки будет равен нулю, так как весь буфер заполнен терминаторами строки.

Пример:

```
void OnStart()
  {
//---
   string str;
   StringInit(str,200,0);
   Print("str = ",str,": StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/*  Результат
str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

Смотри также

StringBufferLen, StringLen

© 2000-2025, MetaQuotes Ltd.

StringLen

Возвращает число символов в строке.

```
int  StringLen(
   string  string_value      // строка
   );
```

Параметры

string_value

[in] Строка для вычисления длины.

Возвращаемое значение

Количество символов в строке без учета завершающего нуля.

Пример:

```
void OnStart()
  {
//--- определяем тестовую строку
   string text="123456789012345";
//--- получаем количество символов в строке
   int str_len=StringLen(text);
//--- распечатаем в журнале строку и количество символов в ней
   PrintFormat("The StringLen() function returned the value of %d chars in string: '%s

   /*
   Результат
   The StringLen() function returned the value of 15 chars in string: '12345678901234
   */
  }
```

Смотри также

StringBufferLen, StringTrimLeft, StringTrimRight, StringToCharArray, StringToShortArray

© 2000-2025, MetaQuotes Ltd.

StringSetLength

Устанавливает для строки указанную длину в символах.

```
bool  StringSetLength(
   string&     string_var,       // строка
   uint        new_length        // новая длина строки
   );
```

Параметры

string_var

[in][out] Строка, для которой необходимо задать новую длину в символах.

new_capacity

[in] Требуемая длина строки в символах. Если новая длина new_length меньше текущего размера, то непоместившиеся символы будут отброшены.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Функция StringSetLength() не изменяет размера буфера, отведенного под строку.

Пример:

```
void OnStart()
  {
//--- определяем строку
   string text="123456789012345";

//--- выводим в журнал строку и её размер
   PrintFormat("Before StringSetLength() the string '%s' has a size of %d characters",

//--- уменьшаем размер строки до 10 символов
   StringSetLength(text, 10);

//--- выводим в журнал строку, изменённую в результате работы StringSetLength(), и её
   PrintFormat("After StringSetLength() the string is now '%s', and has a size of %d c

   /*
   Результат
   Before StringSetLength() the string '123456789012345' has a size of 15 characters
   After StringSetLength() the string is now '1234567890', and has a size of 10 charac
   */
  }
```

Смотри также

StringLen, StringBufferLen, StringReserve StringInit, StringSetCharacter

© 2000-2025, MetaQuotes Ltd.

StringReplace

Заменяет в строке все найденные подстроки на заданную последовательность символов.

```
int  StringReplace(
   string&         str,              // строка, в которой будет осуществляться замена
   const string    find,             // искомая подстрока
   const string    replacement       // подстрока, которая будет вставлена в найденные
   );
```

Параметры

str

[in][out] Строка, в которой необходимо произвести замены.

find

[in] Искомая подстрока для замены.

replacement

[in] Подстрока, которая будет вставлена вместо найденной.

Возвращаемое значение

Количество произведенных замен в случае успеха, в случае ошибки -1. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Если функция успешно отработала, но замены не произведены (не найдена заменяемая подстрока), то возвращается 0.

Причиной ошибки могут быть неверные параметры *str* или *find* (пустая или неинициализированна строка, см. StringInit()). Кроме того, ошибка возникнет, если для завершения замен недостаточно памяти.

Пример:

```
  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);

//  Результат
//  Replaced: 3. Result=The slow black bear jumped over the lazy dog.
//
```

Смотри также

StringSetCharacter(), StringSubstr()

© 2000-2025, MetaQuotes Ltd.

StringReserve

Резервирует в памяти для строки буфер указанного размера.

```
bool  StringReserve(
   string&     string_var,       // строка
   uint        new_capacity      // размер буфера для хранения строки
   );
```

Параметры

string_var

[in][out] Строка, для которой необходимо изменить размер буфера.

new_capacity

[in] Требуемый размер буфера под строку. Если новый размер new_capacity меньше длины строки, то размер текущего буфера не изменится.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

В общем случае размер строки не равен размеру буфера, предназначенному для хранения строки. Как правило при создании строки буфер под неё выделяется с запасом. Функция StringReserve() позволяет управлять размером буфера и указываеть оптимальный размер для будущих операций.

В отличие от StringInit() функция StringReserve() не изменяет содержимое строки и не заполняет её символами.

Пример:

```
void OnStart()
  {
   string s;
//--- проверим скорость работы без использования StringReserve
   ulong t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_no_reserve=GetMicrosecondCount()-t0;
   s=NULL;
//--- теперь замерим с использованием StringReserve
   StringReserve(s,1024 * 3);
   t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_reserve=GetMicrosecondCount()-t0;
//--- проверим время
   Print("Test with StringReserve passed for "+(string)msc_reserve+" msc");
   Print("Test without StringReserve passed for "+(string)msc_no_reserve+" msc");
```

© 2000-2025, MetaQuotes Ltd.

```
/* Результат
     Test with StringReserve passed for 50 msc
     Test without StringReserve passed for 121 msc
*/
  }
```

Смотри также

StringBufferLen, StringSetLength, StringInit, StringSetCharacter

© 2000-2025, MetaQuotes Ltd.

StringSetCharacter

Возвращает копию строки с измененным значением символа в указанной позиции.

```
bool  StringSetCharacter(
   string&   string_var,        // строка
   int       pos,               // позиция
   ushort    character          // символ
   );
```

Параметры

string_var

[in][out] Строка.

pos

[in] Позиция символа в строке. Может быть от 0 до StringLen(text).

character

[in] Символьный код Unicode.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Если значение pos меньше длины строки и значение символьного кода = 0, то строка усекается (но размер буфера, распределенного под строку остается неизменным). Длина строки становится равной значению pos.

Если значение параметра *pos* равняется значению длины строки, то указанный символ добавляется в конец строки, и таким образом длина строки увеличивается на единицу.

Пример:

```
void OnStart()
  {
   string str="0123456789";
   Print("before: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- вставим нулевое значение посреди строки
   StringSetCharacter(str,6,0);
   Print("  after: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- добавим символ в конец строки
   int size=StringLen(str);
   StringSetCharacter(str,size,'+');
   Print("addition: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) =",StringLen(str));
  }
/* Результат
```

© 2000-2025, MetaQuotes Ltd.

```
   before: str = 0123456789 ,StringBufferLen(str) = 0   StringLen(str) = 10
    after: str = 012345 ,StringBufferLen(str) = 16   StringLen(str) = 6
   addition: str = 012345+ ,StringBufferLen(str) = 16   StringLen(str) = 7
*/
```

Смотри также

StringBufferLen, StringLen, StringFill, StringInit, CharToString, ShortToString, CharArrayToString, ShortArrayToString

© 2000-2025, MetaQuotes Ltd.

StringSplit

Получает из указанной строки подстроки по заданному разделителю и возвращает количество полученных подстрок.

```
int  StringSplit(
   const string   string_value,        // строка для поиска подстрок
   const ushort   separator,           // разделитель, по которому в строке будут искат
   string         & result[]           // массив, переданный по ссылке, для получения п
   );
```

Параметры

string_value

[in] Строка, из которой необходимо получить подстроки. Сама строка при этом не изменяется.

pos

[in] Код символа разделителя. Для получения кода можно использовать функцию StringGetCharacter().

result[]

[out] Массив строк, в который помещаются полученные подстроки.

Возвращаемое значение

Количество полученных строк в массиве result[]. Если разделитель в переданной строке не найден, то в массив будет помещена только одна исходная строка.

Если строка string_value пустая или NULL, то функция вернет ноль. В случае ошибки функция вернет -1. Для получения кода ошибки нужно вызвать функцию GetLastError().

Пример:

```
string to_split="_мама_мыла_раму_"; // строка для разбивки на подстроки
   string sep="_";                  // разделитель в виде символа
   ushort u_sep;                    // код символа разделителя
   string result[];                 // массив для получения строк
   //--- получим код разделителя
   u_sep=StringGetCharacter(sep,0);
   //--- разобьем строку на подстроки
   int k=StringSplit(to_split,u_sep,result);
   //--- выведем комментарий
   PrintFormat("Получено строк: %d. Использован разделитель '%s' с кодом %d",k,sep,u_s
   //--- теперь выведем все полученные строки
   if(k>0)
     {
      for(int i=0;i<k;i++)
        {
         PrintFormat("result[%d]=\"%s\"",i,result[i]);
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

StringReplace(), StringSubstr(), StringConcatenate()

© 2000-2025, MetaQuotes Ltd.

StringSubstr

Извлекает подстроку из текстовой строки, начинающейся с указанной позиции.

```
string  StringSubstr(
   string  string_value,      // строка
   int     start_pos,         // с какой позиции начать
   int     length=-1          // длина извлекаемой строки
   );
```

Параметры

string_value

[in] Строка, из которой должна быть извлечена подстрока.

start_pos

[in] Начальная позиция подстроки. Может быть от 0 до StringLen(text) -1.

length=-1

[in] Длина извлекаемой подстроки. Если значение параметра равно -1 либо параметр не задан, то будет извлекаться подстрока, начиная с указанной позиции и до конца строки.

Возвращаемое значение

Копия извлеченной подстроки, если возможно, иначе возвращается пустая строка.

Пример:

```
void OnStart()
  {
//--- получаем наименование текущего символа
   string name = Symbol();

//--- получаем базовую и котируемую валюты символа
   string base   = StringSubstr(name, 0, 3);
   string quoted = StringSubstr(name, 3, 3);

//--- распечатаем в журнале полученные валюты символа
   PrintFormat("Symbol: %s. Currency base: %s, currency quoted: %s", name, base, quote

   /*
   Результат
   Symbol: EURUSD. Currency base: EUR, currency quoted: USD
   */
  }
```

Смотри также

StringSplit, StringFind, StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

StringToLower

Преобразует все символы указанной строки в строчные (маленькие) по месту.

```
bool  StringToLower(
   string&  string_var      // строка для обработки
   );
```

Параметры

string_var

[in][out] Строка.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Пример:

```
void OnStart()
  {
//--- определяем исходную строку в верхнем регистре
   string text=" - THIS STRING, WRITTEN IN UPPERCASE, MUST BE WRITTEN IN LOWERCASE";
//--- Выводим исходную строку в журнал
   Print("Source line:\n", text);
//--- преобразуем все символы строки в нижний регистр и выводим полученный результат в
   if(StringToLower(text))
      Print("The original string after using the StringToLower() function:\n", text);

   /*
   Результат
   Source line:
    - THIS STRING, WRITTEN IN UPPERCASE, MUST BE WRITTEN IN LOWERCASE
   The original string after using the StringToLower() function:
    - this string, written in uppercase, must be written in lowercase
   */
  }
```

Смотри также

StringToUpper, StringTrimLeft, StringTrimRight

© 2000-2025, MetaQuotes Ltd.

StringToUpper

Преобразует все символы указанной строки в прописные (большие) по месту.

```
bool  StringToUpper(
   string&  string_var      // строка для обработки
   );
```

Параметры

string_var

[in][out] Строка.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Пример:

```
void OnStart()
  {
//--- определяем исходную строку в нижнем регистре
   string text=" - this string, written in lowercase, must be written in uppercase";
//--- Выводим исходную строку в журнал
   Print("Source line:\n", text);
//--- преобразуем все символы строки в верхний регистр и выводим полученный результат
   if(StringToUpper(text))
      Print("The original string after using the StringToUpper() function:\n", text);

   /*
   Результат
   Source line:
    - this string, written in lowercase, must be written in uppercase
   The original string after using the StringToUpper() function:
    - THIS STRING, WRITTEN IN LOWERCASE, MUST BE WRITTEN IN UPPERCASE
   */
  }
```

Смотри также

StringToLower, StringTrimLeft, StringTrimRight

© 2000-2025, MetaQuotes Ltd.

StringTrimLeft

Удаляет символы перевода каретки, пробелы и символы табуляции с начала строки до первого значимого символа. Строка модифицируется по месту.

```
int  StringTrimLeft(
   string&  string_var      // строка для обрезки
   );
```

Параметры

string_var

[in][out] Строка, которая должна быть обрезана слева.

Возвращаемое значение

Возвращает количество отрезанных символов.

Пример:

```
void OnStart()
  {
//--- определяем исходную строку с шестью пробелами слева
   string text="      All spaces on the left will be removed from this string";
//--- Выводим исходную строку в журнал
   PrintFormat("Source line:\n'%s'", text);
//--- удаляем слева все пробелы и выводим количество удалённых символов и полученную с
   int num=StringTrimLeft(text);
   PrintFormat("The StringTrimLeft() function removed %d chars from the left side. Now

   /*
   Результат
   Source line:
   '      All spaces on the left will be removed from this string'
   The StringTrimLeft() function removed 6 chars from the left side. Now the line look
   'All spaces on the left will be removed from this string'
   */
  }
```

Смотри также

StringTrimRight, StringToLower, StringToUpper

© 2000-2025, MetaQuotes Ltd.

StringTrimRight

Удаляет символы перевода каретки, пробелы и символы табуляции от последнего значимого символа до конца строки. Строка модифицируется по месту.

```
int  StringTrimRight(
   string&  string_var      // строка для обрезки
   );
```

Параметры

string_var

[in][out] Строка, которая должна быть обрезана справа.

Возвращаемое значение

Возвращает количество отрезанных символов.

Пример:

```
void OnStart()
  {
//--- определяем исходную строку с шестью пробелами справа
   string text="All spaces on the right will be removed from this string      ";
//--- Выводим исходную строку в журнал
   PrintFormat("Source line:\n'%s'", text);
//--- удаляем справа все пробелы и выводим количество удалённых символов и полученную
   int num=StringTrimRight(text);
   PrintFormat("The StringTrimRight() function removed %d chars from the right side. N

   /*
   Результат
   Source line:
   'All spaces on the right will be removed from this string      '
   The StringTrimRight() function removed 6 chars from the right side. Now the line lo
   'All spaces on the right will be removed from this string'
   */
  }
```

Смотри также

StringTrimLeft, StringToLower, StringToUpper

© 2000-2025, MetaQuotes Ltd.

Дата и время

Группа функций, обеспечивающих работу с данными типа datetime (целое число, представляющее собой количество секунд, прошедших с 0 часов 1 января 1970 года).

Для организации счетчиков и таймеров высокого разрешения нужно использовать функцию GetTickCount(), которая выдает значения в миллисекундах.

Функция	Действие
TimeCurrent	Возвращает последнее известное время сервера (время прихода последней котировки) в формате datetime
TimeTradeServer	Возвращает расчетное текущее время торгового сервера
TimeLocal	Возвращает локальное компьютерное время в формате datetime
TimeGMT	Возвращает время GMT формате datetime с учетом перехода на зимнее или летнее время по локальному времени компьютера, на котором запущен клиентский терминал
TimeDaylightSavings	Возвращает признак перехода на летнее /зимнее время
TimeGMTOffset	Возвращает текущую разницу между временем GMT и локальным временем компьютера в секундах с учетом перехода на зимнее или летнее время
TimeToStruct	Производит конвертацию из значения типа datetime в переменную типа структуры MqlDateTime
StructToTime	Производит конвертацию из переменной типа структуры MqlDateTime в значение типа datetime

© 2000-2025, MetaQuotes Ltd.

TimeCurrent

Возвращает последнее известное время сервера, время прихода последней котировки по одному из выбранных в "Обзоре рынка" символов. В обработчике OnTick() данная функция вернет время пришедшего обрабатываемого тика. В других случаях (например, вызов в обработчиках OnInit(), OnDeinit(), OnTimer() и так далее) это - время прихода последней котировки по любому символу, доступного в окне "Обзор рынка", то самое время, которое показано в заголовке этого окна. Значение времени формируется на торговом сервере и не зависит от настроек времени на компьютере пользователя. Существует 2 варианта функции.

Вызов без параметров

```
datetime  TimeCurrent();
```

Вызов с параметром типа MqlDateTime

```
datetime  TimeCurrent(
   MqlDateTime&  dt_struct      // переменная типа структуры
   );
```

Параметры

dt_struct

[out] Переменная типа структуры MqlDateTime.

Возвращаемое значение

Значение типа datetime

Примечание

Если в качестве параметра была передана переменная типа структуры MqlDateTime, то она заполняется соответствующим образом.

Для организации счетчиков и таймеров высокого разрешения нужно использовать функцию GetTickCount(), которая выдает значения в миллисекундах.

При работе в тестере стратегий время последней котировки TimeCurrent() моделируется в соответствии с историческими данными.

Пример:

```
void OnStart()
  {
//--- объявим переменную MqlDateTime для заполнения данными о дате/времени и получим в
   MqlDateTime tm={};
   datetime    time1=TimeCurrent();    // первая форма вызова: время последней котиров
   datetime    time2=TimeCurrent(tm);  // вторая форма вызова: время последней котиров

//--- выведем в журнал результат получения даты/времени и заполнения структуры соответ
   PrintFormat("Tick time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\n-
               (string)time1, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day
   /*
   результат:
```

© 2000-2025, MetaQuotes Ltd.

```
   Tick time: 2024.04.18 15:40:06
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 15
   - Min: 40
   - Sec: 06
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeTradeServer

Возвращает расчетное текущее время торгового сервера. В отличие от функции TimeCurrent(), расчет значения времени производится в клиентском терминале и зависит от настроек времени на компьютере пользователя. Существует 2 варианта функции.

Вызов без параметров

```
datetime  TimeTradeServer();
```

Вызов с параметром типа MqlDateTime

```
datetime  TimeTradeServer(
   MqlDateTime&  dt_struct      // переменная типа структуры
   );
```

Параметры

dt_struct

[out] Переменная типа структуры MqlDateTime.

Возвращаемое значение

Значение типа datetime

Примечание

Если в качестве параметра была передана переменная типа структуры MqlDateTime, то она заполняется соответствующим образом.

Для организации счетчиков и таймеров высокого разрешения нужно использовать функцию GetTickCount(), которая выдает значения в миллисекундах.

При работе в тестере стратегий время TimeTradeServer() моделируется в соответствии с историческими данными и всегда равно TimeCurrent().

Пример:

```
void OnStart()
  {
//--- объявим переменную MqlDateTime для заполнения данными о дате/времени и получим в
   MqlDateTime tm={};
   datetime    time_current=TimeCurrent();                  // первая форма вызова: вр
   datetime    time_server =TimeTradeServer(tm);            // вторая форма вызова: ра
   int         difference  =int(time_current-time_server);  // разница между Time Curr

//--- выведем в журнал время последней котировки и расчетное текущее время торгового с
   PrintFormat("Time Current: %s\nTime Trade Server: %s\n- Year: %u\n- Month: %02u\n-
               "- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of
               (string)time_current, (string)time_server, tm.year, tm.mon, tm.day, tm.
               EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week), difference);
   /*
   результат:
   Time Current: 2024.04.18 16:10:14
```

© 2000-2025, MetaQuotes Ltd.

```
   Time Trade Server: 2024.04.18 16:10:15
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 16
   - Min: 10
   - Sec: 15
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   Difference between Time Current and Time Trade Server: -1
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeLocal

Возвращает локальное время компьютера, на котором запущен клиентский терминал. Существует 2 варианта функции.

Вызов без параметров

```
datetime  TimeLocal();
```

Вызов с параметром типа MqlDateTime

```
datetime  TimeLocal(
   MqlDateTime&  dt_struct      // переменная типа структуры
   );
```

Параметры

dt_struct

[out] Переменная типа структуры MqlDateTime.

Возвращаемое значение

Значение типа datetime

Примечание

Если в качестве параметра была передана переменная типа структуры MqlDateTime, то она заполняется соответствующим образом.

Для организации счетчиков и таймеров высокого разрешения нужно использовать функцию GetTickCount(), которая выдает значения в миллисекундах.

При работе в тестере стратегий локальное время TimeLocal() всегда равно моделируемому серверному времени TimeCurrent().

Пример:

```
void OnStart()
  {
//--- объявим переменную MqlDateTime для заполнения данными о локальном времени компьк
   MqlDateTime tm={};
   datetime    time1=TimeLocal();   // первая форма вызова: локальное время компьютера
   datetime    time2=TimeLocal(tm); // вторая форма вызова: локальное время компьютера

//--- выведем в журнал результат получения локального времени компьютера и заполнения
   PrintFormat("Local time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\r
               (string)time1, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day
   /*
   результат:
   Local time: 2024.04.18 19:44:09
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 19
```

© 2000-2025, MetaQuotes Ltd.

```
   - Min: 44
   - Sec: 09
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeGMT

Возвращает время GMT, которое вычисляется с учётом перехода на зимнее или летнее время по локальному времени компьютера, на котором запущен клиентский терминал. Существует 2 варианта функции.

Вызов без параметров

```
datetime  TimeGMT();
```

Вызов с параметром типа MqlDateTime

```
datetime  TimeGMT(
   MqlDateTime&  dt_struct      // переменная типа структуры
   );
```

Параметры

dt_struct

[out] Переменная типа структуры MqlDateTime.

Возвращаемое значение

Значение типа datetime

Примечание

Если в качестве параметра была передана переменная типа структуры MqlDateTime, то она заполняется соответствующим образом.

Для организации счетчиков и таймеров высокого разрешения нужно использовать функцию GetTickCount(), которая выдает значения в миллисекундах.

При работе в тестере стратегий время TimeGMT() всегда равно моделируемому серверному времени TimeTradeServer().

Пример:

```
void OnStart()
  {
//--- объявим переменную MqlDateTime для заполнения данными о дате/времени и получим з
   MqlDateTime tm={};
   datetime    time1=TimeLocal();            // первая форма вызова: локальное время к
   datetime    time2=TimeGMT(tm);            // вторая форма вызова: время GMT, вычисл
   int         shift=int(time1-time2)/3600;  // смещение локального времени относител

//--- выведем в журнал локальное время и время GMT с данными заполненной структуры Mql
   PrintFormat("Time Local: %s\nTime GMT: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n
               "- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of
               (string)time1, (string)time2, tm.year, tm.mon, tm.day, tm.hour, tm.min,
               EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week), shift);
   /*
   результат:
   Time Local: 2024.04.18 19:37:23
```

© 2000-2025, MetaQuotes Ltd.

```
   Time GMT: 2024.04.18 12:37:23
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 12
   - Min: 37
   - Sec: 23
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   Local time offset relative to GMT: +7
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeDaylightSavings

Возвращает поправку на летнее время в секундах, если был произведен переход на летнее время. Зависит от настроек времени на компьютере пользователя.

```
int  TimeDaylightSavings();
```

Возвращаемое значение

Если был произведен переход на зимнее (стандартное) время, то возвращается 0.

Пример:

```
void OnStart()
  {
//--- получаем поправку на летнее время в секундах
   int sec_dl=TimeDaylightSavings();

//--- создаём текст описания полученного значения
   string text=(sec_dl==0 ? "Standard \"winter\" time is used" :
                StringFormat("Daylight saving time has been switched over. The correct

//--- выводим в журнал описание поправки на летнее время в секундах
   Print(text);
   /*
   результат для "зимнего" времени:
   Standard "winter" time is used

   результат для "летнего" времени:
   Daylight saving time has been switched over. The correction is -3600 seconds
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeGMTOffset

Возвращает текущую разницу между временем GMT и локальным временем компьютера в секундах с учетом перехода на зимнее или летнее время. Зависит от настроек времени на компьютере пользователя.

```
int  TimeGMTOffset();
```

Возвращаемое значение

Значение типа int, представляющее текущую разницу между временем GMT и локальным временем компьютера TimeLocal() и в секундах.

```
TimeGMTOffset() =  TimeGMT() - TimeLocal()
```

Пример:

```
void OnStart()
  {
//--- получаем локальное время, время GMT и разницу между временем GMT и локальным вре
   datetime time_local=TimeLocal();
   datetime time_gmt  =TimeGMT();
   int      offset    =TimeGMTOffset();

//--- покажем в журнале полученные значения времени и смещения
   PrintFormat("Local Time: %s, GMT Time: %s, Seconds Offset: %+d", (string)time_local
   /*
   результат:
   Local Time: 2024.04.18 19:35:52, GMT Time: 2024.04.18 12:35:52, Seconds Offset: -25
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeToStruct

Производит конвертацию из значения типа datetime (количество секунд с 01.01.1970) в переменную типа структуры MqlDateTime.

```
bool  TimeToStruct(
   datetime      dt,            // дата и время
   MqlDateTime&  dt_struct      // структура для принятия значений
   );
```

Параметры

dt

[in] Значение даты для конвертации.

dt_struct

[out] Переменная структуры типа MqlDateTime.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
void OnStart()
  {
//---получим последнее известное время сервера, объявим структуру даты/времени и  запо
   datetime    time=TimeCurrent();
   MqlDateTime tm  ={};
   if(!TimeToStruct(time,tm))
      Print("TimeToStruct() failed. Error ", GetLastError());

//--- выведем в журнал полученное время сервера и результат заполнения структуры MqlDa
   PrintFormat("Server time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\
               (string)time, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day_
   /*
   результат:
   Server time: 2024.04.18 15:47:27
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 15
   - Min: 47
   - Sec: 27
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

StructToTime

Производит конвертацию из переменной типа структуры MqlDateTime в значение типа datetime и возвращает полученное значение.

```
datetime  StructToTime(
   MqlDateTime&  dt_struct      // структура даты и времени
   );
```

Параметры

dt_struct

[in] Переменная структуры типа MqlDateTime.

Возвращаемое значение

Значение типа datetime, содержащее количество секунд с 01.01.1970.

Пример:

```
#property script_show_inputs

input int   InpYear  =  0;    // Year
input int   InpMonth =  0;    // Month
input int   InpDay   =  0;    // Day
input int   InpHour  =  0;    // Hour
input int   InpMin   =  0;    // Minutes
input int   InpSec   =  0;    // Seconds

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- скорректируем введённые значения и запишем их в переменные
   int year =  (InpYear<1970 ? 1970 : InpYear); // Если год введён меньше, чем 1970, т
   int month=  (InpMonth<1 ?  1  :  InpMonth > 12  ?  12 :  InpMonth);
   int day  =  (InpDay  <1 ?  1  :  InpDay   > 31  ?  31 :  InpDay);
   int hour =  (InpHour <0 ?  0  :  InpHour  > 23  ?  23 :  InpHour);
   int min  =  (InpMin  <0 ?  0  :  InpMin   > 59  ?  59 :  InpMin);
   int sec  =  (InpSec  <0 ?  0  :  InpSec   > 59  ?  59 :  InpSec);

//--- выведем введённые значения в журнал
   PrintFormat("Entered date and time: %04u.%02u.%02u %02u:%02u:%02u", InpYear, InpMon

//--- выведем скорректированные введённые значения в журнал
   PrintFormat("Corrected date and time: %04u.%02u.%02u %02u:%02u:%02u", year, month,

//--- запишем входные значения в соответствующие поля структуры
   MqlDateTime time_struct={};
   time_struct.year= year;
```

© 2000-2025, MetaQuotes Ltd.

```
   time_struct.mon = month;
   time_struct.day = day;
   time_struct.hour= hour;
   time_struct.min = min;
   time_struct.sec = sec;

//--- сконвертируем дату и время из структуры в переменную с типом datetime и
   datetime time = StructToTime(time_struct);

//--- выведем результат конвертации из переменной типа структуры MqlDateTime в значени
   Print("Converted date and time: ",TimeToString(time,TIME_DATE|TIME_MINUTES|TIME_SEC
   /*
   результаты, если введены нулевые значения по умолчанию:
   Entered date and time: 0000.00.00 00:00:00
   Corrected date and time: 1970.01.01 00:00:00
   Converted date and time: 1970.01.01 00:00:00

   результаты, если введен неправильный день февраля текущего года:
   Entered date and time: 2024.02.31 00:00:00
   Corrected date and time: 2024.02.31 00:00:00
   Converted date and time: 2024.03.02 00:00:00
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Информация о счете

Функции, возвращающие параметры текущего счета.

Функция	Действие
AccountInfoDouble	Возвращает значение типа double соответствующего свойства счета
AccountInfoInteger	Возвращает значение целочисленного типа (bool,int или long) соответствующего свойства счета
AccountInfoString	Возвращает значение типа string соответствующего свойства счета

© 2000-2025, MetaQuotes Ltd.

AccountInfoDouble

Возвращает значение соответствующего свойства счета.

```
double  AccountInfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Значение может быть одним из значений ENUM_ACCOUNT_INFO_DOUBLE.

Возвращаемое значение

Значение типа double.

Пример:

```
void OnStart()
  {
//--- выведем всю информацию, доступную из функции 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));
  }
```

Смотри также

SymbolInfoDouble, SymbolInfoString, SymbolInfoInteger, PrintFormat

© 2000-2025, MetaQuotes Ltd.

AccountInfoInteger

Возвращает значение соответствующего свойства счета.

```
long  AccountInfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Значение может быть одним из значений ENUM_ACCOUNT_INFO_INTEGER.

Возвращаемое значение

Значение типа long.

Примечание

Свойство должно быть одного из типов bool, int или long.

Пример:

```
void OnStart()
  {
//--- выведем всю информацию, доступную из функции 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)AccountInfoIntege

//--- сообщим о возможности совершения торговых операций
   if(thisAccountTradeAllowed)
      Print("Торговля для данного счета разрешена");
   else
      Print("Торговля для данного счета запрещена!");

//--- выясним - можно ли торговать на данном счету экспертами
   if(EATradeAllowed)
      Print("Торговля советниками для данного счета разрешена");
   else
      Print("Торговля советниками для данного счета запрещена!");

//--- выясним тип счета
   switch(tradeMode)
     {
      case(ACCOUNT_TRADE_MODE_DEMO):
         Print("Это демо счет");
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case(ACCOUNT_TRADE_MODE_CONTEST):
         Print("Это конкурсный счет");
         break;
      default:Print("Это реальный счет!");
     }

//--- выясним режим задания уровня StopOut
   switch(stopOutMode)
     {
      case(ACCOUNT_STOPOUT_MODE_PERCENT):
         Print("Уровень StopOut задается в процентах");
         break;
      default:Print("Уровень StopOut задается в денежном выражении");
     }
  }
```

Смотри также

Информация о счете

© 2000-2025, MetaQuotes Ltd.

AccountInfoString

Возвращает значение соответствующего свойства счета.

```
string  AccountInfoString(
   ENUM_ACCOUNT_INFO_STRING  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Значение может быть одним из значений ENUM_ACCOUNT_INFO_STRING.

Возвращаемое значение

Значение типа string.

Пример:

```
void OnStart()
  {
//--- выведем всю информацию, доступную из функции AccountInfoString()
   Print("Имя брокера = ",AccountInfoString(ACCOUNT_COMPANY));
   Print("Валюта депозита = ",AccountInfoString(ACCOUNT_CURRENCY));
   Print("Имя клиента = ",AccountInfoString(ACCOUNT_NAME));
   Print("Название торгового сервера = ",AccountInfoString(ACCOUNT_SERVER));
  }
```

Смотри также

Информация о счете

© 2000-2025, MetaQuotes Ltd.

Проверка состояния

Функции, возвращающие параметры текущего состояния клиентского терминала

Функция	Действие
GetLastError	Возвращает значение последней ошибки
IsStopped	Возвращает true, если поступила команда завершить выполнение mql5-программы
UninitializeReason	Возвращает код причины деинициализации
TerminalInfoInteger	Возвращает значение целого типа соответствующего свойства окружения mql5-программы
TerminalInfoDouble	Возвращает значение типа double соответствующего свойства окружения mql5-программы
TerminalInfoString	Возвращает значение типа string соответствующего свойства окружения mql5-программы
MQLInfoInteger	Возвращает значение целого типа соответствующего свойства запущенной mql5-программы
MQLInfoString	Возвращает значение типа string соответствующего свойства запущенной mql5-программы
Symbol	Возвращает имя символа текущего графика.
Period	Возвращает значение таймфрейма текущего графика
Digits	Возвращает количество десятичных знаков после запятой, определяющее точность измерения цены символа текущего графика
Point	Возвращает размер пункта текущего инструмента в валюте котировки.

© 2000-2025, MetaQuotes Ltd.

GetLastError

Возвращает содержимое системной переменной _LastError.

```
int  GetLastError();
```

Возвращаемое значение

Возвращает значение последней ошибки, произошедшей во время исполнения mql5-программы.

Примечание

После вызова функции содержимое переменной _LastError не обнуляется. Чтобы обнулить эту переменную, необходимо вызвать функцию ResetLastError()

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[1]={};    // сюда запишем данные текущего бара

//--- намеренно вызываем функцию с неподходящими параметрами
   int res=CopyRates(NULL, PERIOD_CURRENT, 0, 2, rates);
   if(res!=2)
      PrintFormat("CopyRates() returned %d. LastError %d", res, GetLastError());

//--- сбрасываем код последней ошибки перед копированием данных текущего бара в струк
   ResetLastError();
//--- если функция отработает неверно, то код ошибки будет отличаться от 0
   CopyRates(NULL, PERIOD_CURRENT, 0, 1, rates);
   Print("CopyRates() error ", GetLastError());

//--- распечатаем массив полученных значений
   ArrayPrint(rates);
  }
```

Смотри также

Коды возврата торгового сервера

© 2000-2025, MetaQuotes Ltd.

IsStopped

Проверяет принудительное завершение работы mql5-программы.

```
bool  IsStopped();
```

Возвращаемое значение

Возвращает true, если в системной переменной _StopFlag содержится значение, отличное от 0. Ненулевое значение записывается в переменную _StopFlag, если поступила команда завершить выполнение mql5-программы. В этом случае необходимо как можно быстрее завершить работу программы, в противном случае программа будет завершена принудительно извне через 3 секунды.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- в бесконечном цикле с проверкой на стоп
   while(!IsStopped())
     {
      //--- выводим на график локальное время компьютера
      Comment("Time Local: ", TimeToString(TimeLocal(), TIME_DATE|TIME_MINUTES|TIME_SE
      Sleep(16);
     }
   Print("The StopFlag is set. The program will be stopped.");

//--- почистим за собой
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

UninitializeReason

Возвращает код причины деинициализации.

```
int  UninitializeReason();
```

Возвращаемое значение

Возвращает значение переменной _UninitReason, которое формируется перед вызовом функции OnDeinit(). Значение зависит от причины, приведшей к деинициализации.

Пример:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- получаем в переменную reason код причины деинициализации
   int reason=UninitializeReason();
//--- создаём строку сообщения с причиной деинициализации и выводим сообщение в журнал
   string message=StringFormat("%s: Uninitialize reason code: %d (%s)",__FUNCTION__, r
   Print(message);
//--- успешно
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- создаём строку сообщения с кодом причины деинициализации из формальной переменно
   string message=StringFormat("%s: Uninitialize reason code: %d (%s)",__FUNCTION__, r
   Print(message);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
  }
//+------------------------------------------------------------------+
//| Возвращает описание причины деинициализации                      |
//+------------------------------------------------------------------+
string UninitializeReasonDescription(const int reason)
  {
   switch(reason)
     {
      //--- эксперт прекратил свою работу, вызвав функцию ExpertRemove()
      case REASON_PROGRAM :
```

© 2000-2025, MetaQuotes Ltd.

```
        return("Expert Advisor terminated its operation by calling the ExpertRemove()
      //--- программа удалена с графика
      case REASON_REMOVE :
        return("Program has been deleted from the chart");
      //--- программа перекомпилирована
      case REASON_RECOMPILE :
        return("Program has been recompiled");
      //--- символ или период графика был изменен
      case REASON_CHARTCHANGE :
        return("Symbol or chart period has been changed");
      //--- график закрыт
      case REASON_CHARTCLOSE :
        return("Chart has been closed");
      //--- входные параметры были изменены пользователем
      case REASON_PARAMETERS :
        return("Input parameters have been changed by a user");
      //--- активирован другой счет либо произошло переподключение к торговому серверу
      case REASON_ACCOUNT :
        return("Another account has been activated or reconnection to the trade server
      //--- применен другой шаблон графика
      case REASON_TEMPLATE :
        return("A new template has been applied");
      //--- признак того, что обработчик OnInit() вернул ненулевое значение
      case REASON_INITFAILED :
        return("This value means that OnInit() handler has returned a nonzero value");
      //--- терминал был закрыт
      case REASON_CLOSE :
        return("Terminal has been closed");
     }

//--- неизвестная причина деинициализации
   return("Unknown reason");
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoInteger

Возвращает значение соответствующего свойства окружения mql5-программы.

```
int  TerminalInfoInteger(
   int  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Может быть одним из значений перечисления ENUM_TERMINAL_INFO_INTEGER.

Возвращаемое значение

Значение типа int.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем номер билда запущенного терминала и его признак "64-битный терминал"
   int  build = TerminalInfoInteger(TERMINAL_BUILD);
   bool x64   = TerminalInfoInteger(TERMINAL_X64);

//--- распечатаем в журнале полученные данные для терминала
   PrintFormat("MetaTrader 5 %s build %d", (x64 ? "x64" : "x32"), build);
   /*
   результат:
   MetaTrader 5 x64 build 4330
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoDouble

Возвращает значение соответствующего свойства окружения mql5-программы.

```
double  TerminalInfoDouble(
   int  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Может быть одним из значений перечисления ENUM_TERMINAL_INFO_DOUBLE.

Возвращаемое значение

Значение типа double.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем баланс пользователя в MQL5.community
   double balance = TerminalInfoDouble(TERMINAL_COMMUNITY_BALANCE);

//--- выводим полученное значение в журнал
   PrintFormat("Balance in MQL5.community: %.2f", balance);
   /*
   результат
   Balance in MQL5.community: 230.50
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoString

Функция возвращает значение соответствующего свойства окружения mql5-программы. Свойство должно быть типа string

```
string  TerminalInfoString(
   int  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Может быть одним из значений перечисления ENUM_TERMINAL_INFO_STRING.

Возвращаемое значение

Значение типа string.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим данные операционной системы и терминала
   string os_ver  = TerminalInfoString(TERMINAL_OS_VERSION);      // наименование опер
   string name    = TerminalInfoString(TERMINAL_NAME);            // имя терминала
   string path    = TerminalInfoString(TERMINAL_PATH);            // папка, из которой
   string data    = TerminalInfoString(TERMINAL_DATA_PATH);       // папка, в которой
   string common  = TerminalInfoString(TERMINAL_COMMONDATA_PATH); // общая папка всех

//--- выведем полученные данные в журнал
   PrintFormat("OS: %s\nTerminal: %s\n- Path: %s\n- Data path: %s\n- Common Data path:
   /*
   OS: Windows 10 build 19045
   Terminal: MetaTrader 5
   - Path: E:\MetaQuotes\MetaTrader 5
   - Data path: E:\MetaQuotes\MetaTrader 5
   - Common Data path: C:\Users\admin\AppData\Roaming\MetaQuotes\Terminal\Common
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

MQLInfoInteger

Возвращает значение соответствующего свойства запущенной MQL5-программы.

```
int  MQLInfoInteger(
   int  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Может быть одним из значений перечисления ENUM_MQL_INFO_INTEGER.

Возвращаемое значение

Значение типа int.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим значения доступной и потребляемой памяти для MQL-программы
   int limit = MQLInfoInteger(MQL_MEMORY_LIMIT);    // максимально возможный объём дина
   int used  = MQLInfoInteger(MQL_MEMORY_USED);     // размер использованной памяти MQ

//--- выведем полученные значения в журнал
   PrintFormat("Maximum possible amount of dynamic memory for MQL5 program: %d Mb\n"+
               "Memory used by MQL5 program: %d Mb", limit, used);
   /*
   результат
   Maximum possible amount of dynamic memory for MQL5 program: 8388608 Mb
   Memory used by MQL5 program: 2 Mb
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

MQLInfoString

Возвращает значение соответствующего свойства запущенной MQL5-программы.

```
string  MQLInfoString(
   int  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Может быть одним из значений перечисления ENUM_MQL_INFO_STRING.

Возвращаемое значение

Значение типа string.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим данные запущенной программы
   string name = MQLInfoString(MQL_PROGRAM_NAME);  // имя запущенной MQL5-программы
   string path = MQLInfoString(MQL_PROGRAM_PATH);  // путь для данной запущенной прогр

//--- выведем полученные данные в журнал
   PrintFormat("Name of the running MQL program: '%s'\nPath of the running MQL progra
   /*
   результат:
   Name of the running MQL program: 'MQLInfoString'
   Path of the running MQL program: E:\MetaQuotes\MetaTrader 5\MQL5\Scripts\MQLInfoSt
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Symbol

Возвращает имя символа текущего графика.

```
string  Symbol();
```

Возвращаемое значение

Значение системной переменной _Symbol, в которой хранится имя символа текущего графика.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем имя символа текущего графика
   string name = Symbol();

//--- выводим полученные данные в журнал
   PrintFormat("Current chart symbol name: '%s'", name);
   /*
   результат
   Current chart symbol name: 'EURUSD'
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Period

Возвращает значение таймфрейма текущего графика.

```
ENUM_TIMEFRAMES  Period();
```

Возвращаемое значение

Содержимое переменной _Period, в которой хранится значение таймфрейма текущего графика. Значение может быть одним из значений перечисления ENUM_TIMEFRAMES.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем значение таймфрейма текущего графика и его описание
   ENUM_TIMEFRAMES period    = Period();
   string          timeframe = StringSubstr(EnumToString(period), 7);

//--- выводим полученные данные в журнал
   PrintFormat("Current chart timeframe: %s\nTimeframe value: %s (%d)",
               timeframe, EnumToString(period), period);
   /*
   результат:
   Current chart timeframe: H4
   Timeframe value: PERIOD_H4 (16388)
   */
  }
```

Смотри также

PeriodSeconds, Периоды графиков, Дата и время, , Видимость объектов

© 2000-2025, MetaQuotes Ltd.

Digits

Возвращает количество десятичных знаков после запятой, определяющее точность измерения цены символа текущего графика.

```
int  Digits();
```

Возвращаемое значение

Значение переменной _Digits, в которой хранится количество десятичных знаков после запятой, определяющее точность измерения цены символа текущего графика.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем количество десятичных знаков после запятой для символа текущего графика
   int digits = Digits();

//--- выводим полученные данные в журнал
   Print("Number of decimal digits for the current chart symbol: ", digits);
   /*
   результат:
   Number of decimal digits for the current chart symbol: 5
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Point

Возвращает размер пункта текущего инструмента в валюте котировки.

```
double  Point();
```

Возвращаемое значение

Значение переменной _Point, в которой хранится размер пункта текущего инструмента в валюте котировки.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем размер пункта текущего инструмента в валюте котировки
   double point = Point();

//--- выводим полученные данные в журнал
   Print("Point size of the current symbol in the quote currency: ", DoubleToString(po
   /*
   результат:
   Point size of the current symbol in the quote currency: 0.00001
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Обработка событий

В языке MQL5 предусмотрена обработка некоторых предопределенных событий. Функции для обработки этих событий должны быть определены в программе MQL5: имя функции, тип возвращаемого значения, состав параметров (если они есть) и их типы должны строго соответствовать описанию функции-обработчика события.

Именно по типу возвращаемого значения и по типам параметров обработчик событий клиентского терминала идентифицирует функции, обрабатывающие то или иное событие. Если у соответствующей функции указаны иные, не соответствующие нижеследующим описаниям, параметры или указан иной тип возвращаемого значения, то такая функция не будет использоваться для обработки события.

Функция	Описание
OnStart	Вызывается в скрипте при наступлении события Start для выполнения действий, заложенных в скрипт
OnInit	Вызывается в индикаторах и экспертах при наступлении события Init для инициализации запущенной MQL5-программы
OnDeinit	Вызывается в индикаторах и экспертах при наступлении события Deinit для деинициализации запущенной MQL5-программы
OnTick	Вызывается в экспертах при наступлении события NewTick для обработки новой котировки
OnCalculate	Вызывается в индикаторах при наступлении события Calculate для обработки изменения ценовых данных
OnTimer	Вызывается в индикаторах и экспертах при наступлении периодического события Timer, которое с заданным интервалом времени генерируется терминалом
OnTrade	Вызывается в экспертах при наступлении события Trade, которое генерируется при завершении торговой операции на торговом сервере
OnTradeTransaction	Вызывается в экспертах при наступлении события TradeTransaction для обработки результатов выполнения торгового запроса
OnBookEvent	Вызывается в экспертах при наступлении события BookEvent для обработки изменений в стакане заявок
OnChartEvent	Вызывается в индикаторах и экспертах при наступлении события ChartEvent для обработки изменений графика, вызванных действиями пользователя или работой MQL5-программ
OnTester	Вызывается в экспертах при наступлении события Tester для выполнения необходимых действий по окончании тестирования
OnTesterInit	Вызывается в экспертах при наступлении события TesterInit для выполнения необходимых действий перед началом оптимизации

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
OnTesterDeinit	Вызывается в экспертах при наступлении события TesterDeinit для выполнения необходимых действий по окончании оптимизации эксперта
OnTesterPass	Вызывается в экспертах при наступлении события TesterPass для обработки поступления нового фрейма данных во время оптимизации эксперта

Клиентский терминал отсылает возникающие события в соответствующие открытые графики. Также события могут генерироваться графиками (события графика) либо mql5-программами (пользовательские события). Генерацию событий создания и удаления графических объектов на графике можно включать и отключать заданием свойств графика CHART_EVENT_OBJECT_CREATE и CHART_EVENT_OBJECT_DELETE. Каждая mql5-программа и каждый график имеют свою собственную очередь событий, куда складываются все вновь поступающие события.

Программа получает события только от графика, на котором она запущена. Все события обрабатываются одно за другим в порядке поступления. Если в очереди уже есть событие NewTick либо это событие находится в состоянии обработки, то новое событие NewTick в очередь mql5-программы не ставится. Аналогично, если в очереди mql5-программы уже находится событие ChartEvent или такое событие обрабатывается, то новое событие такого типа не ставится в очередь. Обработка событий таймера производится по такой же схеме – если в очереди находится или уже обрабатывается событие Timer, то новое событие таймера не ставится в очередь.

Очереди событий имеют ограниченный, но достаточный размер, поэтому переполнение очереди для корректно написанной программы маловероятно. При переполнении очереди новые события отбрасываются без постановки в очередь.

Крайне не рекомендуется использовать бесконечные циклы для обработки событий. Исключением из этого правила могут быть только скрипты, которые обрабатывают одно единственное событие Start.

Библиотеки не обрабатывают никаких событий.

© 2000-2025, MetaQuotes Ltd.

OnStart

Вызывается в скриптах и сервисах при наступлении события Start. Функция предназначена для однократного выполнения действий, заложенных в программу. Существуют два варианта функции.

Версия с возвратом результата

```
int  OnStart(void);
```

Возвращаемое значение

Значение типа int, которое выводится в закладку "Журнал".

После завершения скрипта в журнале терминала будет создана запись вида "script имя_скрипта removed (result code N)", где N и есть то значение, которое вернула функция OnStart().

После завершения сервиса в журнале терминала будет создана запись вида "service имя_сервиса stopped (result code N)", где N и есть то значение, которое вернула функция OnStart().

Приоритетным является использование вызова OnStart() с возвратом результата выполнения, так как этот способ позволяет не только выполнить скрипт или сервис, но и вернуть код ошибки или другую полезную информацию для анализа результата выполнения работы программы.

Версия без возврата результата оставлена только для совместимости со старыми кодами. Не рекомендуется к использованию

```
void  OnStart(void);
```

Примечание

OnStart() является единственной функцией для обработки событий в скриптах и сервисах, другие события в эти программы не посылаются. В свою очередь событие Start не посылается экспертам и пользовательским индикаторам.

Пример скрипта:

```
//--- макросы для работы с цветом
#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()
  {
//--- установим цвет падающей свечи
   Comment("Установим цвет падающей свечи");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BEAR,GetRandomColor());
   ChartRedraw(); // немедленно обновим график без ожидания нового тика
   Sleep(1000);   // сделаем паузу в 1 секунду, чтобы можно было увидеть изменения
//--- установим цвет растущей свечи
   Comment("Установим цвет растущей свечи");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BULL,GetRandomColor());
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(1000);
//--- установим цвет фона
   Comment("Установим цвет фона");
   ChartSetInteger(0,CHART_COLOR_BACKGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- установим цвет линии Ask
   Comment("Установим цвет линии Ask");
   ChartSetInteger(0,CHART_COLOR_ASK,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- установим цвет линии Bid
   Comment("Установим цвет линии Bid");
   ChartSetInteger(0,CHART_COLOR_BID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- установим цвет падающего бара и окантовки падающей свечи
   Comment("Установим цвет падающего бара и окантовки падающей свечи");
   ChartSetInteger(0,CHART_COLOR_CHART_DOWN,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- установим цвет линии графика и свечей типа "Доджи"
   Comment("Установим цвет линии графика и свечей типа Доджи");
   ChartSetInteger(0,CHART_COLOR_CHART_LINE,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- установим цвет растущего бара и окантовки растущей свечи
   Comment("Установим цвет растущего бара и окантовки растущей  свечи");
   ChartSetInteger(0,CHART_COLOR_CHART_UP,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- установим цвет осей, шкалы и строки OHLC
   Comment("Установим цвет осей, шкалы и строки OHLC");
   ChartSetInteger(0,CHART_COLOR_FOREGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- установим цвет сетки
   Comment("Установим цвет сетки");
   ChartSetInteger(0,CHART_COLOR_GRID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- установим цвет цены Last
   Comment("Установим цвет цены Last");
   ChartSetInteger(0,CHART_COLOR_LAST,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- установим цвет уровней ордеров Stop Loss и Take Profit
   Comment("Установим цвет уровней ордеров Stop Loss и Take Profit");
   ChartSetInteger(0,CHART_COLOR_STOP_LEVEL,GetRandomColor());
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartRedraw();
   Sleep(1000);
//--- установим цвет объемов и уровней открытия позиций
   Comment("Установим цвет объемов и уровней открытия позиций");
   ChartSetInteger(0,CHART_COLOR_VOLUME,GetRandomColor());
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Возвращает цвет, сгенерированный случайным образом               |
//+------------------------------------------------------------------+
color GetRandomColor()
  {
   color clr=(color)GETRGB(XRGB(rand()%255,rand()%255,rand()%255));
   return clr;
  }
```

Смотри также

Функции обработки событий, Выполнение программ, События клиентского терминала

© 2000-2025, MetaQuotes Ltd.

OnInit

Вызывается в индикаторах и экспертах при наступлении события Init. Функция предназначена для инициализации запущенной MQL5-программы. Существуют два варианта функции.

Версия с возвратом результата

```
int  OnInit(void);
```

Возвращаемое значение

Значение типа int, ноль означает успешную инициализацию.

При возврате значения INIT_FAILED советник будет принудительно выгружен с графика.

При возврате значения INIT_FAILED индикатор не будет выгружен с графика. При этом оставшийся на графике индикатор является нерабочим — обработчики событий в индикаторе не вызываются.

Приоритетным является использование вызова OnInit() с возвратом результата выполнения, так как этот способ позволяет не только выполнить инициализацию программы, но и вернуть код ошибки в случае досрочного прекращения программы.

Версия без возврата результата оставлена только для совместимости со старыми кодами. Не рекомендуется к использованию

```
void  OnInit(void);
```

Примечание

Событие Init генерируется сразу после загрузки эксперта или индикатора, для скриптов это событие не генерируется. Функция OnInit() используется для инициализации программы MQL5. Если OnInit() имеет возвращаемое значение типа int, то ненулевой код возврата означает неудачную инициализацию и генерирует событие Deinit с кодом причины деинициализации REASON_INITFAILED.

Функция OnInit() типа void всегда означает удачную инициализацию и не рекомендуется к использованию.

При оптимизации входных параметров эксперта рекомендуется в качестве кода возврата использовать значения из перечисления ENUM_INIT_RETCODE. Эти значения предназначены для организации управления ходом оптимизации, в том числе для выбора наиболее подходящих агентов тестирования. Прямо при инициализации эксперта еще до запуска самого тестирования можно запросить информацию о конфигурации и ресурсах агента (количество ядер, объем свободной памяти и т.д.) с помощью функции TerminalInfoInteger(). На основе полученной информации можно либо разрешить использовать данный агент тестирования, либо отказаться от него при оптимизации данного эксперта.

Идентификатор	Описание
INIT_SUCCEEDED	Инициализация прошла успешно, тестирование эксперта можно продолжать. Этот код означает то же самое, что и нулевое значение - инициализация эксперта в тестере прошла успешно.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
INIT_FAILED	Неудачная инициализация, тестирование нет смысла продолжать из-за неустранимых ошибок. Например, не удалось создать индикатор, необходимый для работы эксперта. Возврат этого значения означает то же самое, что и возврат значения, отличного от нуля, - инициализация эксперта в тестере прошла неудачно.
INIT_PARAMETERS_INCORRECT	Предназначен для обозначения программистом некорректного набора входных параметров, в общей таблице оптимизации строка результата с таким кодом возврата будет подсвечена красным цветом. Тестирование для данного набора параметров эксперта не будет выполняться, агент свободен для получения нового задания. При получении этого значения тестер стратегий гарантированно не будет передавать данное задание другим агентам для повторного выполнения.
INIT_AGENT_NOT_SUITABLE	Ошибок в работе программы при инициализации не возникло, но по каким-то причинам данный агент не подходит для проведения тестирования. Например, недостаточно оперативной памяти, нет поддержки OpenCL и так далее. После возврата этого кода агент больше не будет получать заданий до самого конца данной оптимизации.

Использование OnInit() с возвратом INIT_FAILED/INIT_PARAMETERS_INCORRECT в тестере имеет ряд особенностей, которые необходимо учитывать при оптимизации советников:

- набор параметров, для которого OnInit() вернула INIT_PARAMETERS_INCORRECT считается непригодным для тестирования и не будет использован для получения следующей популяции при генетической оптимизации. Если таких "забракованных" наборов параметров будет слишком много, то это может привести к некорректным результатам поиска оптимальных параметров советника. Алгоритм поиска предполагает, что функция критерия оптимизации является гладкой и не имеет разрывов на всём множестве входных параметров.
- если OnInit() вернула INIT_FAILED, то это означает, что тестирование нельзя начинать и советник выгружается из памяти агента. Для выполнения следующего прохода на новом наборе параметров будет произведена повтороная загрузка советника. Это потребует намного больше времени для запуска тестирования следующего прохода оптимизации, чем в случае вызова TesterStop().

Пример функции OnInit() для эксперта

```
//--- input parameters
input int      ma_period=20; // период скользящей средней

//--- хендл индикатора, который используется в советнике
int indicator_handle;
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- проверим на корректность значение ma_period
   if(ma_period<=0)
     {
      PrintFormat("Недопустимое значение входного параметра ma_period: %d",ma_period);
      return (INIT_PARAMETERS_INCORRECT);
     }
//--- при оптимизации
   if(MQLInfoInteger(MQL_OPTIMIZATION))
     {
      //--- проверим объем доступной оперативной памяти для агента
      int available_memory_mb=TerminalInfoInteger(TERMINAL_MEMORY_TOTAL);
      if(available_memory_mb<2000)
        {
         PrintFormat("Недостаточный объем памяти для агента тестирования: %d MB",
                     available_memory_mb);
         return (INIT_AGENT_NOT_SUITABLE);
        }
     }
//--- проверим наличие индикатора
   indicator_handle=iCustom(_Symbol,_Period,"My_Indicator",ma_period);
   if(indicator_handle==INVALID_HANDLE)
     {
      PrintFormat("Не удалось создать хендл индикатора My_Indicator. Код ошибки %d",
                  GetLastError());
      return (INIT_FAILED);
     }
//--- инициализация эксперта прошла удачно
   return(INIT_SUCCEEDED);
  }
```

Смотри также

OnDeinit, TesterHideIndicators, Функции обработки событий, Выполнение программ, События клиентского терминала, Инициализация переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

OnDeinit

Вызывается в индикаторах и экспертах при наступлении события Deinit. Функция предназначена для деинициализации запущенной MQL5-программы.

```
void  OnDeinit(
   const int  reason          // код причины деинициализации
   );
```

Параметры

reason

[in] Код причины деинициализации.

Возвращаемое значение

Нет возвращаемого значения

Примечание

Событие Deinit генерируется для экспертов и индикаторов в следующих случаях:

- перед переинициализацией в связи со сменой символа или периода графика, к которому прикреплена mql5-программа;
- перед переинициализацией в связи со сменой входных параметров;
- перед выгрузкой mql5-программы.

Параметр *reason* может принимать следующие значения:

Константа	Значение	Описание
REASON_PROGRAM	0	Эксперт прекратил свою работу, вызвав функцию ExpertRemove()
REASON_REMOVE	1	Программа удалена с графика
REASON_RECOMPILE	2	Программа перекомпилирована
REASON_CHARTCHANGE	3	Символ или период графика был изменен
REASON_CHARTCLOSE	4	График закрыт
REASON_PARAMETERS	5	Входные параметры были изменены пользователем
REASON_ACCOUNT	6	Активирован другой счет либо произошло переподключение к торговому серверу вследствие изменения настроек счета
REASON_TEMPLATE	7	Применен другой шаблон графика
REASON_INITFAILED	8	Обработчик OnInit() вернул ненулевое значение
REASON_CLOSE	9	Терминал был закрыт

Коды причины деинициализации эксперта можно также получить функцией UninitializeReason() или из предопределенной переменной _UninitReason.

© 2000-2025, MetaQuotes Ltd.

Пример функций OnInit() и OnDeinit() для эксперта

```
input int fake_parameter=3;      // бесполезный параметр
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Получим номер билда, в котором скомпилирована программа
   Print(__FUNCTION__," Build #",__MQLBUILD__);
//--- Код причины перезагрузки можно получать и в OnInit()
   Print(__FUNCTION__," При перезагрузке эксперта можно получить код причины деинициал
//--- Первый способ получить код причины деинициализации
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Второй способ получить код причины деинициализации
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Первый способ получить код причины деинициализации
   Print(__FUNCTION__," Код причины деинициализации = ",reason);
//--- Второй способ получить код причины деинициализации
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Третий способ получить код причины деинициализации
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
  }
//+------------------------------------------------------------------+
//| Возвращает текстовое описания причины деинициализации            |
//+------------------------------------------------------------------+
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:
```

© 2000-2025, 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;
  }
```

Смотри также

OnInit, Функции обработки событий, Выполнение программ, События клиентского терминала, Причины деинициализации, Область видимости и время жизни переменных, Создание и уничтожение объектов

© 2000-2025, MetaQuotes Ltd.

OnTick

Вызывается в экспертах при наступлении события NewTick для обработки новой котировки.

```
void  OnTick(void);
```

Возвращаемое значение

Нет возвращаемого значения

Примечание

Событие NewTick генерируется только для экспертов при поступлении нового тика по символу, к графику которого прикреплен эксперт. Функцию OnTick() бесполезно определять в пользовательском индикаторе или скрипте, поскольку событие NewTick для них не генерируется.

Событие Tick генерируется только для экспертов, но это не означает, что эксперты обязаны иметь функцию OnTick(), так как для экспертов генерируются не только события NewTick, но и события Timer, BookEvent и ChartEvent.

Все события обрабатываются одно за другим в порядке поступления. Если в очереди уже есть событие NewTick либо это событие находится в состоянии обработки, то новое событие NewTick в очередь mql5-программы не ставится.

Событие NewTick генерируется независимо от того, запрещена или разрешена автоматическая торговля (кнопка "Авто-торговля"). Запрет автоматической торговли означает только запрет на отправку торговых запросов из эксперта, работа эксперта при этом не прекращается.

Запрет автоматической торговли путем нажатия кнопки "Авто-торговля" не прерывает текущее выполнение функции OnTick().

Пример эксперта, в котором вся торговая логика помещена в функцию 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 "Пример советника, торгующего в направлении \"взрывной\" свечи"
#property description "\"Взрывная\" свеча имеет тело размером более k*ATR"
#property description "Параметр \"revers\" переворачивает направление сигнала"

input double lots=0.1;         // объем в лотах
input double kATR=3;           // длина сигнальной свечи в ATR
input int    ATRperiod=20;     // период индикатора ATR
input int    holdbars=8;       // сколько баров удерживаем позицию
input int    slippage=10;      // допустимое проскальзывание
input bool   revers=false;     // переворачиваем сигнал?
input ulong  EXPERT_MAGIC=0;   // MagicNumber эксперта
```

© 2000-2025, MetaQuotes Ltd.

```
//--- для хранения хендла индикатора ATR
int atr_handle;
//--- здесь будем хранить последние значения ATR и тела свечи
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- инициализируем глобальные переменные
   last_atr=0;
   last_body=0;
//--- установим правильный объем
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- создадим хендл индикатора ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: не удалось создать iATR, код ошибки %d",__FUNCTION__,GetLastEr
      return(INIT_FAILED);
     }
//--- успешная инициализация эксперта
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- сообщим код завершения работы эксперта
   Print(__FILE__,": Код причины деинициализации = ",reason);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- торговый сигнал
   static int signal=0; // +1 означает сигнал на покупку, -1 означает сигнал на прода
//--- проверим и закроем старые позиции, открытые более holdbars баров назад
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- проверим появление нового бара
   if(isNewBar())
     {
      //--- проверим наличие сигнала
      signal=CheckSignal();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- если открыта неттинговая позиция, то сигнал пропускаем - ждем, пока она закроет
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // выходим из обработчика события NewTick и не входим в рынок до появле
     }
//--- для хеджингового счета каждая позиция живет и закрывается раздельно
   if(signal!=0)
     {
      //--- сигнал на покупку
      if(signal>0)
        {
         PrintFormat("%s: Есть сигнал на покупку! Revers=%s",__FUNCTION__,string(reve
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- сигнал на продажу
      if(signal<0)
        {
         PrintFormat("%s: Есть сигнал на продажу! Revers=%s",__FUNCTION__,string(reve
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- конец функции OnTick
  }
//+------------------------------------------------------------------+
//| Проверяет наличие торгового сигнала                              |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 означает отсутствие сигнала
   int res=0;
//--- получим значение ATR на предпоследнем завершенном баре (индекс бара равен 2)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- получим данные последнего закрытого бара в массив типа MqlRates
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- вычислим размер тела бара на последнем закрытом баре
         last_body=bar[0].close-bar[0].open;
         //--- если тело последнего бара (с индексом 1) превышает предыдущее значение
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // для растущей свечи положительное значение
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      else
         PrintFormat("%s: Не удалось получить последний бар! Ошибка",__FUNCTION__,GetL
     }
   else
      PrintFormat("%s: Не удалось получить значение индикатора ATR! Ошибка",__FUNCTION
//--- если включен реверсивный режим торговли
   res=revers?-res:res;  // если нужно, то развернем сигнал (вместо 1 вернем -1, а вме
//--- вернем значение торгового сигнала
   return (res);
  }
//+------------------------------------------------------------------+
//|  Возвращает true при появлении нового бара                       |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // храним время открытия текущего бара
//--- получим время открытия нулевого бара
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- если время открытия изменилось, значит появился новый бар
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- нужно ли выводить в лог информацию о времени открытия нового бара
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- выведем сообщение о времени открытия нового бара
         PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- получим данные о последнем тике
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() failed, error = ",GetLastError());
         //--- выведем время последнего тика с точностью до миллисекунд
         PrintFormat("Last tick was at %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- у нас есть новый бар
      return (true);
     }
//--- нового бара нет
   return (false);
  }
//+------------------------------------------------------------------+
//| Покупка по рынку с заданным объемом                              |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- покупаем по рыночной цене
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Продажа по рынку с заданным объемом                              |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- продаем по рыночной цене
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Закрытие позиций по времени удержания в барах                    |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // количество открытых позиций
//--- перебор всех открытых позиций
   for(int i=total-1; i>=0; i--)
     {
      //--- параметры позиции
      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;

      //--- если позиция живет уже долго, а также MagicNumber и символ совпадают
      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); // понижаем регистр текста для правильного форматиро
         PrintFormat("Закрываем позицию #%I64u %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- установка типа ордера и отправки торгового запроса
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Подготовка и отправка торгового запроса                          |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- объявление и инициализация структур
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- параметры запроса
   request.action   =TRADE_ACTION_DEAL;                     // тип торговой операции
   request.position =pos_ticket;                            // тикет позиции, если за
   request.symbol   =Symbol();                              // символ
   request.volume   =volume;                                // объем
   request.type     =type;                                  // тип ордера
   request.price    =price;                                 // цена совершения сделки
   request.deviation=slip;                                  // допустимое отклонение
   request.magic    =magicnumber;                           // MagicNumber ордера
//--- отправка запроса
   if(!OrderSend(request,result))
     {
      //--- выведем информацию о неудаче
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- сообщим об успешной операции
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Смотри также

Функции обработки событий, Выполнение программ, События клиентского терминала, OnTimer, OnBookEvent, OnChartEvent

© 2000-2025, MetaQuotes Ltd.

OnCalculate

Вызывается в индикаторах при наступлении события Calculate для обработки изменений ценовых данных. Существуют два варианта функции, в пределах одного индикатора нельзя использовать оба варианта.

Вычисление на основе массива данных

```
int  OnCalculate(
   const int        rates_total,       // размер массива price[]
   const int        prev_calculated,   // количество обработанных баров на предыдущем
   const int        begin,             // номер индекса в массиве price[], с которого
   const double&    price[]            // массив значений для расчета
   );
```

Вычисления на основе таймсерий текущего таймфрейма

```
int  OnCalculate(
   const int        rates_total,       // размер входных таймсерий
   const int        prev_calculated,   // количество обработанных баров на предыдущем
   const datetime&  time[],            // массив Time
   const double&    open[],            // массив Open
   const double&    high[],            // массив High
   const double&    low[],             // массив Low
   const double&    close[],           // массив Close
   const long&      tick_volume[],     // массив Tick Volume
   const long&      volume[],          // массив Real Volume
   const int&       spread[]           // массив Spread
   );
```

Параметры

rates_total

[in] Размер массива price[] или входных таймсерий, доступных индикатору для расчета. Во втором варианте функции значение параметра соответствует количеству баров на графике, на котором он запущен.

prev_calculated

[in] Содержит значение, которое вернула функция OnCalculate() на предыдущем вызове. Предназначено для пропуска в расчетах тех баров, которые не изменились с предыдущего запуска этой функции.

begin

[in] Значение индекса в массиве price[], с которого начинаются значимые данные. Позволяет пропустить в расчетах отсутствующие или начальные данные, для которых нет корректных значений.

price[]

[in] Массив значений для проведения вычислений. В качестве массива price[] может быть передана одна из ценовых таймсерий либо рассчитанный буфер какого-либо индикатора. Тип

© 2000-2025, MetaQuotes Ltd.

данных, которые были переданы на расчет, можно узнать с помощью предопределенной переменной _AppliedTo.

time{}

[in] Массив со значениями времени открытия баров.

open[]

[in] Массив со значениями цен открытия.

high[]

[in] Массив со значениями максимальных цен.

low[]

[in] Массив со значениями минимальных цен.

close[]

[in] Массив со значениями цен закрытия.

tick_volume[]

[in] Массив со значениями тиковых объемов.

volume[]

[in] Массив со значениями торговых объемов.

spread[]

[in] Массив со значениями спреда для баров.

Возвращаемое значение

Значение типа int, которое будет передано в качестве параметра *prev_calculated* при последующем вызове функции.

Примечание

Если функция OnCalculate() возвращает нулевое значение, то в окне DataWindow клиентского терминала значения индикатора не показываются.

Если с момента последнего вызова функции OnCalculate() ценовые данные были изменены (была загружена более глубокая история или были заполнены пропуски истории), то значение входного параметра *prev_calculated* будет установлено в нулевое значение самим терминалом.

Чтобы определить направление индексации в массивах *time[]*, *open[]*, *high[]*, *low[]*, *close[]*, *tick_volume[]*, *volume[]* и *spread[]*, необходимо вызывать функцию ArrayGetAsSeries(). Чтобы не зависеть от умолчаний, необходимо безусловно вызывать функцию ArraySetAsSeries() для тех массивов, с которыми предполагается работать.

При использовании первого варианта функции выбор необходимой таймсерии или индикатора в качестве массива price[] осуществляется пользователем на вкладке Parameters при запуске индикатора. Для этого необходимо указать нужный элемент в выпадающем списке поля "Apply to".

Для получения значений пользовательского индикатора из других mql5-программ используется функция iCustom(), возвращающая хэндл индикатора для последующих операций. При этом также можно указать необходимый массив *price[]* или хэндл другого индикатора. Этот параметр должен передаваться последним в списке входных переменных пользовательского индикатора.

© 2000-2025, MetaQuotes Ltd.

Необходимо использовать связь между значением, возвращаемым функцией OnCalculate(), и вторым входным параметром *prev_calculated*. Параметр *prev_calculated* при вызове функции содержит значение, которое вернула функция OnCalculate() на предыдущем вызове. Это позволяет реализовать экономные алгоритмы расчета пользовательского индикатора с тем, чтобы избежать повторных расчетов для тех баров, которые не изменились с предыдущего запуска этой функции.

Пример индикатора

```
//+------------------------------------------------------------------+
//|                                          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 "Пример вычисления индикатора Momentum"

//---- indicator settings
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_type1   DRAW_LINE
#property indicator_color1  Blue
//---- входные параметры
input int MomentumPeriod=14; // Период для расчета
//---- индикаторный буфер
double    MomentumBuffer[];
//--- глобальная переменная для хранения периода расчетов
int       IntPeriod;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- проверим входной параметр
   if(MomentumPeriod<0)
     {
      IntPeriod=14;
      Print("параметр Период имеет неправльное значение. Для расчетов будет использова
     }
   else
      IntPeriod=MomentumPeriod;
//---- буферы
   SetIndexBuffer(0,MomentumBuffer,INDICATOR_DATA);
//---- имя индикатора для показа в DataWindow и в подокне
   IndicatorSetString(INDICATOR_SHORTNAME,"Momentum"+"("+string(IntPeriod)+")");
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим номер бара, с которого будет идти отрисовка
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,IntPeriod-1);
//--- установим 0.0 в качестве пустого значения, которое не отрисовывается
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//--- с какой точностью показывать значения индикатора
   IndicatorSetInteger(INDICATOR_DIGITS,2);
  }
//+------------------------------------------------------------------+
//|  Расчет индикатора Momentum                                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,     // размер массива price[]
                const int prev_calculated, // сколько баров обработано ранее
                const int begin,           // откуда начинаются значимые данные
                const double &price[])     // массив для значения для обработки
  {
//--- стартовая позиция для вычислений
   int StartCalcPosition=(IntPeriod-1)+begin;
//---- если недостаточно данных для расчета
   if(rates_total<StartCalcPosition)
      return(0);  // выходим с нулевым значением - индикатор не рассчитан
//--- correct draw begin
   if(begin>0)
      PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,StartCalcPosition+(IntPeriod-1));
//--- начинаем расчеты, вычислим позицию начала
   int pos=prev_calculated-1;
   if(pos<StartCalcPosition)
      pos=begin+IntPeriod;
//--- основной цикл вычислений
   for(int i=pos;i<rates_total && !IsStopped();i++)
      MomentumBuffer[i]=price[i]*100/price[i-IntPeriod];
//--- выполенение OnCalculate завершено. Вернем новое значение prev_calculated для пос
   return(rates_total);
  }
```

Смотри также

ArrayGetAsSeries, ArraySetAsSeries, iCustom, Функции обработки событий, Выполнение программ, События клиентского терминала, Доступ к таймсериям и индикаторам

© 2000-2025, MetaQuotes Ltd.

OnTimer

Вызывается в экспертах при наступлении события Timer, генерируемого терминалом с заданным интервалом.

```
void  OnTimer(void);
```

Возвращаемое значение

Нет возвращаемого значения

Примечание

Событие Timer периодически генерируется клиентским терминалом для эксперта, который активизировал таймер при помощи функции EventSetTimer(). Обычно эта функция вызывается в функции OnInit(). При завершении работы эксперта необходимо уничтожить созданный таймер при помощи EventKillTimer(), которую обычно вызывают в функции OnDeinit().

Каждый эксперт и каждый индикатор работают со своим таймером и получают события только от него. При завершении работы MQL5-программы таймер уничтожается принудительно, если он был создан, но не был отключен функцией EventKillTimer().

Если необходимо получать события таймера чаще, чем один раз в секунду, используйте EventSetMillisecondTimer() для создания таймера высокого разрешения.

В общем случае при уменьшении периода таймера увеличивается время тестирования, так как возрастает количество вызовов обработчика событий таймера. При работе в режиме реального времени события таймера генерируются не чаще 1 раза в 10-16 миллисекунд, что связано с аппаратными ограничениями.

Для каждой программы может быть запущено не более одного таймера. Каждая MQL5-программа и каждый график имеют свою собственную очередь событий, куда складываются все вновь поступающие события. Если в очереди уже есть событие Timer либо это событие находится в состоянии обработки, то новое событие Timer в очередь mql5-программы не ставится.

Пример эксперта с обработчиком 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 "Пример использования таймера для вычисления времени торгового с
#property description "Советник лучше всего запустить в конце торговой недели перед вы
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- создадим таймер с периодом в 1 секунду
   EventSetTimer(1);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- уничтожим таймер при завершении работы
   EventKillTimer();

  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- время первого вызова OnTimer()
   static datetime start_time=TimeCurrent();
//--- время торгового сервера при первом вызове OnTimer();
   static datetime start_tradeserver_time=0;
//--- вычисляемое время торгового сервера
   static datetime calculated_server_time=0;
//--- локальное время на компьютере
   datetime local_time=TimeLocal();
//--- текущее расчетное время торгового сервера
   datetime trade_server_time=TimeTradeServer();
//--- если по какой-то причине время сервера неизвестно, то выходим досрочно
   if(trade_server_time==0)
      return;
//--- если стартовое значение торгового сервера еще не установлено
   if(start_tradeserver_time==0)
     {
      start_tradeserver_time=trade_server_time;
      //--- установим вычисляемое значение торгового сервера
      Print(trade_server_time);
      calculated_server_time=trade_server_time;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- увеличим время первого вызова 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
//--- выведем значения всех счетчиков на график
   Comment(com);
  }
```

Смотри также

EventSetTimer, EventSetMillisecondTimer, EventKillTimer, GetTickCount, GetMicrosecondCount, События клиентского терминала

© 2000-2025, MetaQuotes Ltd.

OnTrade

Вызывается в экспертах при наступлении события Trade. Функция предназначеня для обработки изменений в списках ордеров, позиций и сделок.

```
void  OnTrade(void);
```

Возвращаемое значение

Нет возвращаемого значения

Примечание

OnTrade() вызывается только для экспертов, в индикаторах и скриптах она игнорируется, даже если добавить в них функцию с таким именем и типом.

При любом торговом действии (выставлении отложенного ордера, открытии/закрытии позиции, установке стопов, срабатывании отложенных ордеров и т.п.) соответствующим образом изменяется история ордеров и сделок и/или список позиций и текущих ордеров.

В момент обработки ордера торговый сервер посылает терминалу сообщение о наступлении торгового события Trade. Для получения из истории актуальных данных по ордерам и сделкам необходимо предварительно выполнить запрос торговой истории с помощью HistorySelect().

Торговые события генерируются сервером в следующих случаях:

- изменение в действующих ордерах,
- изменения в позициях,
- изменения в сделках,
- изменения в торговой истории.

Каждое событие Trade может быть результатом одного или нескольких торговых запросов. Торговые запросы отправляются на сервер с помощью OrderSend() или OrderSendAsync(). Каждый запрос может порождать несколько торговых событий. Нельзя полагаться на правило "Один запрос - Одно событие Trade", так как обработка запросов может происходить в несколько этапов и каждая операция может изменять состояние ордеров, позиций и торговой истории.

Обработчик OnTrade() вызывается после соответствующих вызовов OnTradeTransaction(). В общем случае нет точного соотношения по количеству вызовов OnTrade() и OnTradeTransaction(). Один вызов OnTrade() соответствует одному или нескольким вызовам OnTradeTransaction.

Пример эксперта с обработчиком 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"
```

© 2000-2025, MetaQuotes Ltd.

```
input    int days=7;             // глубина торговой истории в днях
//--- зададим на глобальном уровне границы торговой истории
datetime     start;              // дата начала торговой истории в кэше
datetime     end;                // дата конца торговой истории в кэше
//--- глобальные счетчики
int          orders;             // количество действующих ордеров
int          positions;          // количество открытых позиций
int          deals;              // количество сделок в кэше торговой истории
int          history_orders;     // количество ордеров в кэше торговой истории
bool         started=false;      // флаг актуальности счетчиков

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   end=TimeCurrent();
   start=end-days*PeriodSeconds(PERIOD_D1);
   PrintFormat("Границы загружаемой торговой истории: начало - %s, конец - %s",
               TimeToString(start),TimeToString(end));
   InitCounters();
//---
   return(0);
  }
//+------------------------------------------------------------------+
//|  инициализация счетчиков позиций, ордеров и сделок               |
//+------------------------------------------------------------------+
void InitCounters()
  {
   ResetLastError();
//--- загрузим историю
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Не удалось загрузить в кэш историю с %s по %s. Код ошибки: %d",
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- получим текущие значения
   orders=OrdersTotal();
   positions=PositionsTotal();
   deals=HistoryDealsTotal();
   history_orders=HistoryOrdersTotal();
   started=true;
   Print("Счетчики ордеров, позиций и сделок успешно инициализированы");
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| вызывается при поступлении события Trade                         |
//+------------------------------------------------------------------+
void OnTrade()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| пример обработки изменений в торговле и истории                  |
//+------------------------------------------------------------------+
void SimpleTradeProcessor()
  {
   end=TimeCurrent();
   ResetLastError();
//--- загрузим в кэш программы торговую историю из указанного интервала
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Не удалось загрузить в кэш историю с %s по %s. Код ошибки: %d",
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- получим текущие значения
   int curr_orders=OrdersTotal();
   int curr_positions=PositionsTotal();
   int curr_deals=HistoryDealsTotal();
   int curr_history_orders=HistoryOrdersTotal();
//--- проверим изменения в количестве действующих ордеров
   if(curr_orders!=orders)
     {
      //--- количество действующих ордеров изменилось
      PrintFormat("Изменилось количество ордеров. Было %d, стало %d",
                  orders,curr_orders);
      //--- обновим значение
      orders=curr_orders;
     }
//--- изменения в количестве открытых позиций
   if(curr_positions!=positions)
     {
      //--- количество открытых позиций изменилось
      PrintFormat("Изменилось количество позиций. Было %d, стало %d",
```

© 2000-2025, MetaQuotes Ltd.

```
                  positions,curr_positions);
      //--- обновим значение
      positions=curr_positions;
     }
//--- изменения в количестве сделок в кэше торговой истории
   if(curr_deals!=deals)
     {
      //--- количество сделок в кэше торговой истории изменилось
      PrintFormat("Изменилось количество сделок. Было %d, стало %d",
                  deals,curr_deals);
      //--- обновим значение
      deals=curr_deals;
     }
//--- изменения в количестве исторических ордеров в кэше торговой истории
   if(curr_history_orders!=history_orders)
     {
      //--- количество исторических ордеров в кэше торговой истории изменилось
      PrintFormat("Изменилось количество ордеров в истории. Было %d, стало %d",
                  history_orders,curr_history_orders);
     //--- обновим значение
     history_orders=curr_history_orders;
     }
//--- проверка на необходимость изменения границ торговой истории для запроса в кэш
   CheckStartDateInTradeHistory();
  }
//+------------------------------------------------------------------+
//|  изменения начальной даты для запроса торговой истории           |
//+------------------------------------------------------------------+
void CheckStartDateInTradeHistory()
  {
//--- начальный интервал, если бы мы начали работу прямо сейчас
   datetime curr_start=TimeCurrent()-days*PeriodSeconds(PERIOD_D1);
//--- убедимся, что граница начала торговой истории ушла не больше
//--- чем на 1 день от задуманной даты
   if(curr_start-start>PeriodSeconds(PERIOD_D1))
     {
      //--- придется подкорректировать дату начала загружаемой в кэш истории
      start=curr_start;
      PrintFormat("Новая граница начала загружаемой торговой истории: начало => %s",
                  TimeToString(start));
      //--- теперь заново загрузим торговую историю для обновленного интервала
      HistorySelect(start,end);
      //--- подкорректируем счетчики сделок и ордеров в истории для следующего сравнен
      history_orders=HistoryOrdersTotal();
      deals=HistoryDealsTotal();
     }
  }
//+------------------------------------------------------------------+
/* Пример вывода:
```

© 2000-2025, MetaQuotes Ltd.

```
   Границы загружаемой торговой истории: начало - 2018.07.16 18:11, конец - 2018.07.23
   Счетчики ордеров, позиций и сделок успешно инициализированы
   Изменилось количество ордеров. Было 0, стало 1
   Изменилось количество ордеров. Было 1, стало 0
   Изменилось количество позиций. Было 0, стало 1
   Изменилось количество сделок. Было 0, стало 1
   Изменилось количество ордеров в истории. Было 0, стало 1
*/
```

Смотри также

OrderSend, OrderSendAsync, OnTradeTransaction, События клиентского терминала

© 2000-2025, MetaQuotes Ltd.

OnTradeTransaction

Вызывается в экспертах при наступлении события TradeTransaction. Функция предназначена для обработки результатов выполнения торгового запроса.

```
void  OnTradeTransaction()
   const MqlTradeTransaction&    trans,     // структура торговой транзакции
   const MqlTradeRequest&        request,   // структура запроса
   const MqlTradeResult&         result     // структура ответа
   );
```

Параметры

trans

[in] Переменная типа MqlTradeTransaction с описанием транзакции, проведенной на торговом счете.

request

[in] Переменная типа MqlTradeRequest с описанием торгового запроса, породившего транзакцию. Содержит значения только для транзакции типа TRADE_TRANSACTION_REQUEST.

result

[in] Переменная типа MqlTradeResult с результатом выполнения торгового запроса, породившего транзакцию. Содержит значения только для транзакции типа TRADE_TRANSACTION_REQUEST.

Возвращаемое значение

Нет возвращаемого значения

Примечание

OnTradeTransaction() вызывается для обработки события TradeTransaction, которое торговый сервер посылает терминалу в следующих случаях:

- отправка торгового запроса из MQL5-программы при помощи функций OrderSend()/OrderSendAsync() и его последующее исполнение;
- отправка торгового запроса вручную через графический интефейс и его последующее исполнение;
- срабатывания отложенных ордеров и стоп-ордеров на сервере;
- выполнения операций на стороне торгового сервера.

Информация о типе транзакции содержится в поле *type* переменной *trans*. Типы торговых транзакций описываются в перечислении ENUM_TRADE_TRANSACTION_TYPE:

- TRADE_TRANSACTION_ORDER_ADD - добавление нового действующего ордера
- TRADE_TRANSACTION_ORDER_UPDATE - изменение действующего ордера
- TRADE_TRANSACTION_ORDER_DELETE - удаление ордера из списка действующих
- TRADE_TRANSACTION_DEAL_ADD - добавление сделки в историю
- TRADE_TRANSACTION_DEAL_UPDATE - изменение сделки в истории
- TRADE_TRANSACTION_DEAL_DELETE - удаление сделки из истории
- TRADE_TRANSACTION_HISTORY_ADD - добавление ордера в историю в результате исполнения или отмены

© 2000-2025, MetaQuotes Ltd.

- TRADE_TRANSACTION_HISTORY_UPDATE - изменение ордера, находящегося в истории ордеров
- TRADE_TRANSACTION_HISTORY_DELETE - удаление ордера из истории ордеров
- TRADE_TRANSACTION_POSITION - изменение позиции, не связанное с исполнением сделки
- TRADE_TRANSACTION_REQUEST - уведомление о том, что торговый запрос обработан сервером и результат его обработки получен.

При обрабокте транзакций типа TRADE_TRANSACTION_REQUEST для получения дополнительной информации необходимо анализировать второй и третий параметры функции OnTradeTransaction() - *request* и *result*.

Отправка торгового запроса на покупку приводит к цепи торговых транзакций, которые совершаются на торговом счете: 1) запрос принимается на обработку, 2) далее для счета создается соответствующий ордер на покупку, 3) затем происходит исполнение ордера, 4) удаление исполненного ордера из списка действующих, 5) добавление в историю ордеров, 6) далее добавляется соответствующая сделка в историю и 7) создается новая позиция. Все эти действия являются торговыми транзакциями. Приход каждой такой транзакции в терминал является событием TradeTransaction. При этом очередность поступления этих транзакций в терминал не гарантирована, поэтому нельзя свой торговый алгоритм строить на ожидании поступления одних торговых транзакций после прихода других.

Во время обработки торговых транзакций экспертом при помощи обработчика OnTradeTransaction(), терминал продолжает обрабатывать вновь поступающие торговые транзакции. Таким образом, состояние торгового счета может измениться уже в процессе работы OnTradeTransaction(). Например, пока MQL5-программа обрабатывает событие добавления нового ордера, он может быть исполнен, удален из списка открытых и перемещен в историю. В дальнейшем программа будет уведомлена о всех этих событиях.

Длина очереди транзакций составляет 1024 элемента. Если OnTradeTransaction() будет обрабатывать очередную транзакцию слишком долго, старые транзакции в очереди могут быть вытеснены более новыми.

Обработчик OnTrade() вызывается после соответствующих вызовов OnTradeTransaction(). В общем случае нет точного соотношения по количеству вызовов OnTrade() и OnTradeTransaction(). Один вызов OnTrade() соответствует одному или нескольким вызовам OnTradeTransaction.

Каждое событие Trade может быть результатом одного или нескольких торговых запросов. Торговые запросы отправляются на сервер с помощью OrderSend() или OrderSendAsync(). Каждый запрос может порождать несколько торговых событий. Нельзя полагаться на правило "Один запрос - Одно событие Trade", так как обработка запросов может происходить в несколько этапов и каждая операция может изменять состояние ордеров, позиций и торговой истории.

Пример эксперта с обработчиком 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"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Пример слушателя событий 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;   // счетчик вызовов OnTradeTransaction()
   static uint lasttime=0; // время последнего вызова OnTradeTransaction()
//---
   uint time=GetTickCount();
//--- если последняя транзакция была больше 1 секунды назад
   if(time-lasttime>1000)
     {
      counter=0; // значит, это новая торговая операция и можно сбросить счетчик
      if(IS_DEBUG_MODE)
         Print(" Новая торговая операция");
     }
   lasttime=time;
   counter++;
   Print(counter,". ",__FUNCTION__);
//--- результат выполнения торгового запроса
   ulong            lastOrderID   =trans.order;
   ENUM_ORDER_TYPE  lastOrderType =trans.order_type;
   ENUM_ORDER_STATE lastOrderState=trans.order_state;
//--- имя символа, по которому произошла транзакция
   string trans_symbol=trans.symbol;
//--- тип транзакции
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TRADE_TRANSACTION_TYPE  trans_type=trans.type;
   switch(trans.type)
     {
      case  TRADE_TRANSACTION_POSITION:    // изменение позиции
        {
         ulong pos_ID=trans.position;
         PrintFormat("MqlTradeTransaction: Position  #%I64u %s modified: SL=%.5f TP=%.
                     pos_ID,trans_symbol,trans.price_sl,trans.price_tp);
        }
      break;
      case TRADE_TRANSACTION_REQUEST:      // отправка торгового запроса
         PrintFormat("MqlTradeTransaction: TRADE_TRANSACTION_REQUEST");
         break;
      case TRADE_TRANSACTION_DEAL_ADD:     // добавление сделки
        {
         ulong           lastDealID   =trans.deal;
         ENUM_DEAL_TYPE lastDealType =trans.deal_type;
         double         lastDealVolume=trans.volume;
         //--- идентификатор сделки во внешней системе - тикет, присваиваемый биржей
         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 #%I64u %s %s %.2f lot   %s",EnumToS
                     lastDealID,EnumToString(lastDealType),trans_symbol,lastDealVolume
        }
      break;
      case TRADE_TRANSACTION_HISTORY_ADD: // добавление ордера в историю
        {
         //--- идентификатор ордера во внешней системе - тикет, присваиваемый биржей
         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 #%I64u %s %s %s   %s",EnumToStrin
                     lastOrderID,EnumToString(lastOrderType),trans_symbol,EnumToStrin
        }
      break;
      default: // прочие транзакции
        {
         //--- идентификатор ордера во внешней системе - тикет, присваиваемый Московск
         string Exchange_ticket="";
         if(lastOrderState==ORDER_STATE_PLACED)
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             if(OrderSelect(lastOrderID))
                Exchange_ticket=OrderGetString(ORDER_EXTERNAL_ID);
             if(Exchange_ticket!="")
                Exchange_ticket=StringFormat("Exchange ticket=%s",Exchange_ticket);
            }
          PrintFormat("MqlTradeTransaction: %s order #%I64u %s %s   %s",EnumToString(ti
                      lastOrderID,EnumToString(lastOrderType),EnumToString(lastOrderSta
        }
      break;
     }
//--- тикет ордера
   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
//---
  }
//+------------------------------------------------------------------+
//| переводит числовые коды ответов в строковые мнемокоды            |
//+------------------------------------------------------------------+
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;
```

© 2000-2025, MetaQuotes Ltd.

```
      case 10029: return("TRADE_RETCODE_FROZEN");              break;
      case 10030: return("TRADE_RETCODE_INVALID_FILL");        break;
      case 10031: return("TRADE_RETCODE_CONNECTION");          break;
      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;
     }
//---
  }
```

Смотри также

OrderSend, OrderSendAsync, OnTradeTransaction, Структура торгового запроса, Структура торговой транзакции, Типы торговых транзакций, Типы торговых операций, События клиентского терминала

© 2000-2025, MetaQuotes Ltd.

OnBookEvent

Вызывается в индикаторах и экспертах при наступлении события BookEvent. Функция предназначена для обработки изменений стакана цен (Depth of Market).

```
void  OnBookEvent(
   const string&  symbol          // символ
   );
```

Параметры

symbol

[in] Имя финансового инструмента, по которому пришло событие BookEvent

Возвращаемое значение

Нет возвращаемого значения

Примечание

Чтобы получать события BookEvent по любому символу, достаточно предварительно подписаться на получение этих событий для этого символа с помощью функции MarketBookAdd(). Для того чтобы отписаться от получения события BookEvent по конкретному символу, необходимо вызывать функцию MarketBookRelease().

Событие BookEvent является широковещательным в пределах графика. Это означает, что достаточно одному приложению на графике подписаться на получение события BookEvent с помощью функции MarketBookAdd, как все остальные индикаторы и эксперты, запущенные на этом графике и имеющие обработчик OnBookEvent(), будут получать это событие. Поэтому необходимо анализировать имя символа, которое передается в обработчик OnBookEvent() в качестве параметра *symbol*.

Для всех приложений, запущенных на одном графике, ведутся отдельные счетчики на получение событий BookEvent в разрезе символов. Это означает, что на каждом графике может быть несколько подписок на разные символы, и для каждого символа ведется свой собственный счетчик. Подписка и отписка событий BookEvent изменяют счетчик подписок только указанных символов, но при этом только в пределах одного графика. Это означает, что на двух соседних графиках могут быть подписки на события BookEvent на один и тот же символ, но с разными значениями счетчиков подписок.

Начальное значение счетчика подписок равно нулю. При каждом вызове MarketBookAdd() счетчик подписок для указанного символа на данном графике увеличивается на единицу (символ графика и символ в MarketBookAdd() не обязаны совпадать). При вызове MarketBookRelease() счетчик подписок на указанный символ в пределах графика уменьшается на единицу. Трансляция событий BookEvent по любому символу в пределах графика продолжается до тех пор, пока счетчик подписок по данному символу не станет равным нулю. Поэтому важно, чтобы каждая MQL5-программа, которая содержит вызовы MarketBookAdd(), при завершении своей работы правильно отписывалась от получения событий по каждому использованному символу с помощью MarketBookRelease(). Для этого достаточно, чтобы количество вызовов MarketBookAdd() и MarketBookRelease() по каждому вызову было четным за всё время жизни MQL5-программы. Использование флагов или собственных счетчиков подписок внутри программы позволяет безопасно работать с событиями BookEvent и предотвращает отключение подписок на получение этого события в чужих программах в пределах одного графика.

© 2000-2025, MetaQuotes Ltd.

События BookEvent никогда не пропускаются и всегда ставятся в очередь, даже если в данный момент еще не закончена обработка предыдущего события BookEvent. При этом необходимо иметь в виду, что события BookEvent доставляются сами по себе и не несут с собой состояния стакана заявок. Это означает, что вызов MarketBookGet() из обработчика OnBookEvent() позволяет получить текущее актуальное состояние стакана на момент вызова, а не то состояние стакана, которое вызвало отправку события BookEvent. Для гарантированного получения всех уникальных состояний стакана функция OnBookEvent() должна быть максимально быстрой.

Пример

```
//+------------------------------------------------------------------+
//|                                          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/ru/articles/2635"
#property version   "1.00"
#property description "Пример замера скорости обновления стакана с помощью OnBookEvent
#property description "Код взят из статьи https://www.mql5.com/ru/articles/2635"
//--- входные параметры
input ulong ExtCollectTime   =30;  // время теста в секундах
input ulong ExtSkipFirstTicks=10;  // количество пропускаемых тиков на старте
//--- флаг наличия подписки на получение событий BookEvent
bool book_subscribed=false;
//--- массив для приема заявок из стакана
MqlBookInfo  book[];
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- покажем старт
   Comment(StringFormat("Ждем поступления первых %I64u тиков",ExtSkipFirstTicks));
   PrintFormat("Ждем поступления первых %I64u тиков",ExtSkipFirstTicks);
//--- включим трансляцию стакана
   if(MarketBookAdd(_Symbol))
     {
      book_subscribed=true;
      PrintFormat("%s: Функция MarketBookAdd(%s) вернула true",__FUNCTION__,_Symbol);
     }
   else
      PrintFormat("%s: Функция MarketBookAdd(%s) вернула false! GetLastError()=%d",__F
//--- успешная инициализация
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Deinitialize expert                                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- выведем код причины деинициализации
   Print(__FUNCTION__,": Код причины деинициализации = ",reason);
//--- отменим свою подписку на получение событий стакана
   if(book_subscribed)
     {
      if(!MarketBookRelease(_Symbol))
         PrintFormat("%s: MarketBookRelease(%s) вернула false! GetLastError()=%d",_Sy
      else
         book_subscribed=false;
     }
//---
  }
//+------------------------------------------------------------------+
//| BookEvent function                                               |
//+------------------------------------------------------------------+
void OnBookEvent(const string &symbol)
  {
   static ulong starttime=0;              // время начала теста
   static ulong tickcounter=0;            // счетчик обновлений стакана цен
//--- работаем с событиями стакана только в том случае, если мы сами на него подписали
   if(!book_subscribed)
      return;
//--- считаем обновления только по своему символу
   if(symbol!=_Symbol)
      return;
//--- пропускаем первые тики для первичной очистки очереди и разогрева
   tickcounter++;
   if(tickcounter<ExtSkipFirstTicks)
      return;
//--- запомним время старта
   if(tickcounter==ExtSkipFirstTicks)
      starttime=GetMicrosecondCount();
//--- запросим данные стакана
   MarketBookGet(symbol,book);
//--- когда надо остановиться?
   ulong endtime=GetMicrosecondCount()-starttime;
   ulong ticks  =1+tickcounter-ExtSkipFirstTicks;
// сколько прошло времени в микросекундах с начала теста?
   if(endtime>ExtCollectTime*1000*1000)
     {
      PrintFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec ",ticks,endtime/1000.0
      ExpertRemove();
      return;
     }
//--- вывод счетчиков в поле комментария
```

© 2000-2025, MetaQuotes Ltd.

```
   if(endtime>0)
      Comment(StringFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec ",ticks,endti
  }
```

Смотри также

MarketBookAdd, MarketBookRelease, MarketBookGet, OnTrade, OnTradeTransaction, OnTick, Функции обработки событий, Выполнение программ, События клиентского терминала

© 2000-2025, MetaQuotes Ltd.

OnChartEvent

Вызывается в экспертах и индикаторах при наступлении события ChartEvent. Функция предназначена для обработки изменений графика, вызванных действиями пользователя или работой MQL5-программ.

```
void  OnChartEvent()
   const int       id,       // идентификатор события
   const long&     lparam,   // параметр события типа long
   const double&   dparam,   // параметр события типа double
   const string&   sparam    // параметр события типа string
   );
```

Параметры

id

[in] Идентификатор события из перечисления ENUM_CHART_EVENT.

lparam

[in] Параметр события типа long

dparam

[in] Параметр события типа double

sparam

[in] Параметр события типа string

Возвращаемое значение

Нет возвращаемого значения

Примечание

Существуют 11 видов событий, которые можно обрабатывать с помощью предопределенной функции OnChartEvent(). Для пользовательских событий предусмотрено 65535 идентификаторов в диапазоне от CHARTEVENT_CUSTOM до CHARTEVENT_CUSTOM_LAST включительно. Для генерации пользовательского события необходимо использовать функцию EventChartCustom().

Краткое описание событий из перечисления ENUM_CHART_EVENT:

- CHARTEVENT_KEYDOWN — нажатие клавиатуры, когда окно графика находится в фокусе;
- CHARTEVENT_MOUSE_MOVE — перемещение мыши и нажатия кнопок мыши (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE=true);
- CHARTEVENT_OBJECT_CREATE — создание графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_CREATE=true);
- CHARTEVENT_OBJECT_CHANGE — изменение свойств объекта через диалог свойств;
- CHARTEVENT_OBJECT_DELETE — удаление графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_DELETE=true);
- CHARTEVENT_CLICK — щелчок мыши на графике;
- CHARTEVENT_OBJECT_CLICK — щелчок мыши на графическом объекте, принадлежащем графику;
- CHARTEVENT_OBJECT_DRAG — перемещение графического объекта при помощи мыши;

© 2000-2025, MetaQuotes Ltd.

- CHARTEVENT_OBJECT_ENDEDIT — окончание редактирования текста в поле ввода графического объекта Edit (OBJ_EDIT);
- CHARTEVENT_CHART_CHANGE — изменения графика;
- CHARTEVENT_CUSTOM+n — идентификатор пользовательского события, где n находится в диапазоне от 0 до 65535. CHARTEVENT_CUSTOM_LAST содержит последний допустимый идентификатор пользовательского события (CHARTEVENT_CUSTOM+65535).

Все MQL5-программы работают в потоках, отличных от главного потока приложения. Главный поток терминала отвечает за обработку всех системных сообщений Windows, и в результате этой обработки в свою очередь порождает сообщения Windows для своего же приложения. Например, перемещение мышки на графике (событие WM_MOUSE_MOVE) порождает несколько системных сообщений для последующей отрисовки окна приложения, а также посылает внутренние сообщения экспертам и индикаторам, запущенным на этом графике. При этом может возникнуть ситуация, что главный поток приложения ещё не успел обработать системное сообщение WM_PAINT (и поэтому ещё не отрисовал изменённый график), а эксперт или индикатор уже получили событие о перемещении курсора мышки. Тогда в этом случае свойство графика CHART_FIRST_VISIBLE_BAR будет изменено только после отрисовки графика.

Для каждого типа события входные параметры функции OnChartEvent() имеют определенные значения, которые необходимы для обработки этого события. В таблице перечислены события и значения, которые передаются через параметры.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
Событие нажатия клавиатуры	CHARTEVENT_KEYDOWN	код нажатой клавиши	Количество нажатий клавиши, сгенерированных за время ее удержания в нажатом состоянии	Строковое значение битовой маски, описывающее статус кнопок клавиатуры
События мышки (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	X-координата	Y-координата	Строковое значение битовой маски, описывающее статус кнопок мыши
Событие колесика мышки (если для графика установлено свойство CHART_EVENT_MOUSE_WHEEL=true)	CHARTEVENT_MOUSE_WHEEL	Флаги состояний клавиш и кнопок мышки, координаты X и Y курсора. Описание дано в примере	Значение Delta прокрутки колесика мышки	—

© 2000-2025, MetaQuotes Ltd.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
Событие создания графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_CREATE =true)	CHARTEVENT_OBJECT_CREATE	—	—	Имя созданного графического объекта
Событие изменения свойств объекта через диалог свойств	CHARTEVENT_OBJECT_CHANGE	—	—	Имя измененного графического объекта
Событие удаления графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_DELETE =true)	CHARTEVENT_OBJECT_DELETE	—	—	Имя удаленного графического объекта
Событие щелчка мышки на графике	CHARTEVENT_CLICK	X-координата	Y-координата	—
Событие щелчка мышки на графическом объекте	CHARTEVENT_OBJECT_CLICK	X-координата	Y-координата	Имя графического объекта, на котором произошло событие
Событие перемещения графического объекта при помощи мышки	CHARTEVENT_OBJECT_DRAG	—	—	Имя перемещенного графического объекта
Событие окончания редактирования текста в поле	CHARTEVENT_OBJECT_ENDEDIT	—	—	Имя графического объекта "Поле ввода", в

© 2000-2025, MetaQuotes Ltd.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
ввода графического объекта "Поле ввода"				котором завершилось редактирование текста
Событие изменения размеров графика или изменения свойств графика через диалог свойств	CHARTEVENT_CHART_CHANGE	—	—	—
Пользовательское событие с номером N	CHARTEVENT_CUSTOM+N	Значение, заданное функцией EventChartCustom()	Значение, заданное функцией EventChartCustom()	Значение, заданное функцией EventChartCustom()

Пример слушателя событий графика:

```
//+------------------------------------------------------------------+
//|                                          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 "Пример слушателя событий графика и генератора пользовательских
//--- идентификаторы служебных клавиш
#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()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выведем значение константы CHARTEVENT_CUSTOM
   Print("CHARTEVENT_CUSTOM=",CHARTEVENT_CUSTOM);
//---
   Print("Запущен эксперт с именем ",MQLInfoString(MQL5_PROGRAM_NAME));
//--- установка флага получения событий создания объектов графика
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- установка флага получения событий удаления объектов графика
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- включение сообщений о прокрутке колесика мышки
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- принудительное обновление свойств графика гарантирует готовность к обработке соб
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- счетчик тиков для генерации пользовательского события
   static int tick_counter=0;
//--- будем делить накопленные тики на это число
   int simple_number=113;
//---
   tick_counter++;
//--- отправляем пользовательское событие, если счетчик тиков кратен simple_number
   if(tick_counter%simple_number==0)
     {
      //--- сформируем идентификатор пользовательского события в диапазоне от 0 до 655
      ushort custom_event_id=ushort(tick_counter%65535);
      //---  отправим пользовательское событие с заполнением параметров
      EventChartCustom(ChartID(),custom_event_id,tick_counter,SymbolInfoDouble(Symbol
      //--- сделаем вывод в лог для изучения и анализа результатов примера
      Print(__FUNCTION__,": Отправлено пользовательское событие ID=",custom_event_id);
     }
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- нажатие кнопки на клавиатуре
   if(id==CHARTEVENT_KEYDOWN)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      switch((int)lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("Нажата KEY_NUMLOCK_LEFT");   break;
         case KEY_LEFT:          Print("Нажата KEY_LEFT");           break;
         case KEY_NUMLOCK_UP:    Print("Нажата KEY_NUMLOCK_UP");     break;
         case KEY_UP:            Print("Нажата KEY_UP");             break;
         case KEY_NUMLOCK_RIGHT: Print("Нажата KEY_NUMLOCK_RIGHT");  break;
         case KEY_RIGHT:         Print("Нажата KEY_RIGHT");          break;
         case KEY_NUMLOCK_DOWN:  Print("Нажата KEY_NUMLOCK_DOWN");   break;
         case KEY_DOWN:          Print("Нажата KEY_DOWN");           break;
         case KEY_NUMPAD_5:      Print("Нажата KEY_NUMPAD_5");       break;
         case KEY_NUMLOCK_5:     Print("Нажата KEY_NUMLOCK_5");      break;
         default:                Print("Нажата какая-то неперечисленная клавиша");
        }
     }
//--- нажатие левой кнопкой мышки на графике
   if(id==CHARTEVENT_CLICK)
      Print("Координаты щелчка мышки на графике: x = ",lparam,"  y = ",dparam);
//--- нажатие мышкой на графическом объекте
   if(id==CHARTEVENT_OBJECT_CLICK)
      Print("Нажатие кнопки мышки на объекте с именем '"+sparam+"'");
//--- удален объект
   if(id==CHARTEVENT_OBJECT_DELETE)
      Print("Удален объект с именем ",sparam);
//--- создан объект
   if(id==CHARTEVENT_OBJECT_CREATE)
      Print("Создан объект с именем ",sparam);
//--- изменен объект
   if(id==CHARTEVENT_OBJECT_CHANGE)
      Print("Изменен объект с именем ",sparam);
//--- перемещен объект или изменены координаты точек привязки
   if(id==CHARTEVENT_OBJECT_DRAG)
      Print("Изменение точек привязки объекта с именем ",sparam);
//--- изменен текст в поле ввода графического объекта Edit
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      Print("Изменен текст в объекте Edit ",sparam,"  id=",id);
//--- события перемещения мышки
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- разберем состояние кнопок и колесика мышки для этого события
      int flg_keys = (int)(lparam>>32);          // флаг состояний клавиш Ctrl, Shift
      int x_cursor = (int)(short)lparam;         // X-координата, в которой произошло
      int y_cursor = (int)(short)(lparam>>16);   // Y-координата, в которой произошло
      int delta    = (int)dparam;                // суммарное значение прокрутки колес
      //--- обработаем флаг
      string str_keys="";
      if((flg_keys&0x0001)!=0)
```

© 2000-2025, MetaQuotes Ltd.

```
         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_
     }
//--- изменение размеров графика или изменение свойств графика через диалог свойств
   if(id==CHARTEVENT_CHART_CHANGE)
      Print("Изменение размеров или свойств графика");
//--- пользовательское событие
   if(id>CHARTEVENT_CUSTOM)
      PrintFormat("Пользовательское событие ID=%d, lparam=%d, dparam=%G, sparam=%s",id
  }
//+------------------------------------------------------------------+
//| 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);
  }
```

Смотри также

EventChartCustom, Типы событий графика, Функции обработки событий, Выполнение программ, События клиентского терминала

© 2000-2025, MetaQuotes Ltd.

OnTester

Вызывается в экспертах при наступлении события Tester для выполнения необходимых действий по окончании тестирования.

```
double  OnTester(void);
```

Возвращаемое значение

Значение пользовательского критерия оптимизации для оценки результатов тестирования.

Примечание

Функция OnTester() может быть использована только в экспертах при тестировании и предназначена в первую очередь для расчета некоторого значения, используемого в качестве критерия "Custom max" при оптимизации входных параметров.

При генетической оптимизации сортировка результатов в пределах одного поколения производится по убыванию. Это означает, что лучшими с точки зрения критерия оптимизации считаются результаты с наибольшим значением. Худшие значения при такой сортировке помещаются в конец и впоследствии отбрасываются и не принимают участия в формировании следующего поколения.

Таким образом, с помощью функции OnTester() можно не только создавать и сохранять собственные отчеты результатов тестирования, но и управлять ходом оптимизации для поиска наилучших параметров торговой стратегии.

Пример расчета пользовательского критерия оптимизации. Идея заключается в вычислении линейной регрессии графика баланса и описана в статье Оптимизируем стратегию по графику баланса и сравниваем результаты с критерием "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 "Пример советника с обработчиком OnTester()"
#property description "В качестве пользовательского критерия оптимизации "
#property description "возвращается коэффициент линейной регрессии графика баланса,"
#property description "деленный на среднеквадратичную ошибку отклонения"
//--- подключим класс торговых операций
#include <Trade\Trade.mqh>
//--- входные параметры эксперта
input double Lots                = 0.1;      // Объем
input int    Slippage            = 10;       // Допустимое проскальзывание
input int    MovingPeriod        = 80;       // Период скользящей средней
input int    MovingShift         = 6;        // Сдвиг скользящей средней
//--- глобальные переменные
int    IndicatorHandle=0;  // хендл индикатора
```

© 2000-2025, MetaQuotes Ltd.

```
bool   IsHedging=false;     // признак счета
CTrade trade;               // для проведения торговых операций
//---
#define EA_MAGIC 18052018
//+------------------------------------------------------------------+
//| Проверка условий на открытие позиции                             |
//+------------------------------------------------------------------+
void CheckForOpen(void)
  {
   MqlRates rt[2];
//--- торгуем только в начале нового бара
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
//--- тиковый объем
   if(rt[1].tick_volume>1)
      return;
//--- получим значения скользящей средней
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- проверим наличие сигнала
   ENUM_ORDER_TYPE signal=WRONG_VALUE;
//--- свеча открылась выше, а закрылась ниже скользящей средней
   if(rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=ORDER_TYPE_BUY;    // сигнал на покупку
   else // свеча открылась ниже, а закрылась выше скользящей средней
     {
      if(rt[0].open<ma[0] && rt[0].close>ma[0])
         signal=ORDER_TYPE_SELL;// сигнал на продажу
     }
//--- дополнительные проверки
   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);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Проверка условий на закрытие позиции                             |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void CheckForClose(void)
  {
   MqlRates rt[2];
//--- торгуем только в начале нового бара
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
   if(rt[1].tick_volume>1)
      return;
//--- получим значения скользящей средней
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- позиция уже была выбрана ранее с помощью PositionSelect()
   bool signal=false;
   long type=PositionGetInteger(POSITION_TYPE);
//--- свеча открылась выше, а закрылась ниже скользящей средней - закрываем короткую п
   if(type==(long)POSITION_TYPE_SELL && rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=true;
//--- свеча открылась ниже, а закрылась выше скользящей средней - закрываем длинную по
   if(type==(long)POSITION_TYPE_BUY && rt[0].open<ma[0] && rt[0].close>ma[0])
      signal=true;
//--- дополнительные проверки
   if(signal)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
         trade.PositionClose(_Symbol,Slippage);
     }
//---
  }
//+------------------------------------------------------------------+
//| Выбираем позицию с учетом типа счета: Netting или Hedging        |
//+------------------------------------------------------------------+
bool SelectPosition()
  {
   bool res=false;
//--- выбор позиции для счета Hedging
   if(IsHedging)
     {
      uint total=PositionsTotal();
      for(uint i=0; i<total; i++)
        {
         string position_symbol=PositionGetSymbol(i);
```

© 2000-2025, MetaQuotes Ltd.

```
         if(_Symbol==position_symbol && EA_MAGIC==PositionGetInteger(POSITION_MAGIC))
           {
            res=true;
            break;
           }
        }
     }
//--- выбор позиции для счета Netting
   else
     {
      if(!PositionSelect(_Symbol))
         return(false);
      else
         return(PositionGetInteger(POSITION_MAGIC)==EA_MAGIC); //---проверка Magic nur
     }
//--- результат выполнения
   return(res);
  }
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- установим тип торговли: Netting или Hedging
   IsHedging=((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)==ACCOU
//--- инициализируем объект для правильного контроля позиций
   trade.SetExpertMagicNumber(EA_MAGIC);
   trade.SetMarginMode();
   trade.SetTypeFillingBySymbol(Symbol());
   trade.SetDeviationInPoints(Slippage);
//--- создадим индикатор Moving Average
   IndicatorHandle=iMA(_Symbol,_Period,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE);
   if(IndicatorHandle==INVALID_HANDLE)
     {
      printf("Ошибка при создании индикатора iMA");
      return(INIT_FAILED);
     }
//--- ok
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick(void)
  {
//--- если позиция уже открыта, то проверим условие на закрытие
   if(SelectPosition())
      CheckForClose();
// проверим условие на открытие позиции
```

© 2000-2025, MetaQuotes Ltd.

```
   CheckForOpen();
//---
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- значение пользовательского критерия оптимизации (чем больше, тем лучше)
   double ret=0.0;
//--- получим результаты трейдов в массив
   double array[];
   double trades_volume;
   GetTradeResultsToArray(array,trades_volume);
   int trades=ArraySize(array);
//--- если трейдов меньше 10, то тестирование не дало положительных результатов
   if(trades<10)
      return (0);
//--- средний результат на трейд
   double average_pl=0;
   for(int i=0;i<ArraySize(array);i++)
      average_pl+=array[i];
   average_pl/=trades;
//--- выведем сообщение для режима одиночного тестирования
   if(MQLInfoInteger(MQL_TESTER) && !MQLInfoInteger(MQL_OPTIMIZATION))
      PrintFormat("%s: Трейдов=%d, Средняя прибыль=%.2f",__FUNCTION__,trades,average_p
//--- посчитаем коэффициенты линейной регрессии для графика прибыли
   double a,b,std_error;
   double chart[];
   if(!CalculateLinearRegression(array,chart,a,b))
      return (0);
//--- вычислим ошибку отклонения графика от линии регрессии
   if(!CalculateStdError(chart,a,b,std_error))
      return (0);
//--- вычислим отношение трендовой прибыли к среднеквадратичному отклонению
   ret=(std_error == 0.0) ? a*trades : a*trades/std_error;
//--- вернем значение пользовательского критерия оптимизации
   return(ret);
  }
//+------------------------------------------------------------------+
//| Получить массив прибылей/убытков из сделок                       |
//+------------------------------------------------------------------+
bool GetTradeResultsToArray(double &pl_results[],double &volume)
  {
//--- запросим полную торговую историю
   if(!HistorySelect(0,TimeCurrent()))
      return (false);
   uint total_deals=HistoryDealsTotal();
   volume=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим начальный размер массива с запасом - по количеству сделок в истории
   ArrayResize(pl_results,total_deals);
//--- счетчик сделок, фиксирующих торговый результат - прибыль или убыток
   int counter=0;
   ulong ticket_history_deal=0;
//--- пройдем по всем сделкам
   for(uint i=0;i<total_deals;i++)
     {
      //--- выберем сделку
      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
         //--- нас интересуют только торговые операции
         if((deal_type!=DEAL_TYPE_BUY) && (deal_type!=DEAL_TYPE_SELL))
            continue;
         //--- только сделки с фиксацией прибыли/убытка
         if(deal_entry!=DEAL_ENTRY_IN)
           {
            //--- запишем торговый результат в массив и увеличим счетчик трейдов
            pl_results[counter]=deal_profit;
            volume+=deal_volume;
            counter++;
           }
        }
     }
//--- установим окончательный размер массива
   ArrayResize(pl_results,counter);
   return (true);
  }
//+------------------------------------------------------------------+
//| Вычисляет линейную регрессию вида y=a*x+b                        |
//+------------------------------------------------------------------+
bool CalculateLinearRegression(double  &change[],double &chartline[],
                               double  &a_coef,double  &b_coef)
  {
//--- проверим достаточность данных
   if(ArraySize(change)<3)
      return (false);
//--- создадим массив графика с накоплением
   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];
//--- теперь вычислим коэффициенты регрессии
   double x=0,y=0,x2=0,xy=0;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<N;i++)
     {
      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);
  }
//+------------------------------------------------------------------+
//|  Вычисляет среднеквад. ошибку отклонения для заданных a и b      |
//+------------------------------------------------------------------+
bool  CalculateStdError(double  &data[],double  a_coef,double  b_coef,double &std_err)
  {
//--- сумма квадратов ошибки
   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);
  }
```

Смотри также

Тестирование торговых стратегий, TesterHideIndicators, Работа с результатами оптимизации, TesterStatistics, OnTesterInit, OnTesterDeinit, OnTesterPass, MQL_TESTER, MQL_OPTIMIZATION, FileOpen, FileWrite, FileLoad, FileSave

© 2000-2025, MetaQuotes Ltd.

OnTesterInit

Вызывается в экспертах при наступлении события TesterInit для выполнения необходимых действий перед началом оптимизации в тестере стратегий. Существуют два варианта функции.

Версия с возвратом результата

```
int  OnTesterInit(void);
```

Возвращаемое значение

Значение типа int, ноль означает успешную инициализацию эксперта, запущенного на графике перед началом оптимизации.

Приоритетным является использование вызова OnTesterInit() с возвратом результата выполнения, так как этот способ позволяет не только выполнить инициализацию программы, но и вернуть код ошибки в случае досрочного прекращения оптимизации. Возврат любого значения, отличного от INIT_SUCCEEDED (0), означает ошибку и оптимизация запущена не будет.

Версия без возврата результата оставлена только для совместимости со старыми кодами. Не рекомендуется к использованию

```
void  OnTesterInit(void);
```

Примечание

Событие TesterInit автоматически генерируется перед началом оптимизации эксперта в тестере стратегий. По данному событию эксперт, имеющий обработчик OnTesterDeinit() и/или OnTesterPass(), автоматически загружается на отдельном графике терминала с указанными в тестере символом и периодом.

Такой эксперт получает события TesterInit, TesterDeinit и TesterPass, но не получает событий Init, Deinit и NewTick. Соответственно, всю необходимую логику по обработке результатов каждого прохода в процессе оптимизации необходимо реализовать в обработчиках OnTesterInit(), OnTesterDeinit() и OnTesterPass().

Результат каждого одиночного прохода при оптимизации стратегии можно передать через фрейм из обработчика OnTester() с помощью функции FrameAdd().

Функция OnTesterInit() предназначена для инициализации эксперта перед началом оптимизации для последующей обработки результатов оптимизации. Всегда используется совместно с обработчиком OnTesterDeinit().

На выполнение OnTesterInit() отводится ограниченное время, по превышении которого будет произведено принудительное завершение работы эксперта, а сама оптимизация будет отменена. При этом в Журнал тестера будет выведено сообщение:

```
TesterOnTesterInit works too long. Tester cannot be initialized.
```

Пример взят из OnTick, добавлен обработчик OnTesterInit() для установки параметров оптимизации:

```
//+------------------------------------------------------------------+
//|                                          OnTesterInit_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
```

© 2000-2025, 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 "Пример советника с обработчиком OnTesterInit()"
#property description "в котором устанавливаются зачения и границы "
#property description "входных параметров при оптимизации"

input double lots=0. 1;        // объем в лотах
input double kATR=3;           // длина сигнальной свечи в ATR
input int    ATRperiod=20;     // период индикатора ATR
input int    holdbars=8;       // сколько баров удерживаем позицию
input int    slippage=10;      // допустимое проскальзывание
input bool   revers=false;     // переворачиваем сигнал?
input ulong  EXPERT_MAGIC=0;   // MagicNumber эксперта
//--- для хранения хендла индикатора ATR
int atr_handle;
//--- здесь будем хранить последние значения ATR и тела свечи
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//--- запоминаем время начала оптимизации
datetime optimization_start;
//--- для вывода на график длительности после окончании оптимизации
string report;
//+------------------------------------------------------------------+
//| TesterInit function                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- установим значения входных параметров для оптимизации
   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("Установлены начальные значения и границы параметров оптимизации");
//--- запомним начало оптимизации
   optimization_start=TimeLocal();
   report=StringFormat("%s: оптимизация запущена в %s",
                       __FUNCTION__,TimeToString(TimeLocal(),TIME_MINUTES|TIME_SECONDS
//--- выведем собщения на график и в журнал терминала
   Print(report);
   Comment(report);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- продолжительность оптимизации
   string log_message=StringFormat("%s: оптимизация заняла %d секунды",
                                   __FUNCTION__,TimeLocal()-optimization_start);
   PrintFormat(log_message);
   report=report+"\r\n"+log_message;
   Comment(report);
  }
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- инициализируем глобальные переменные
   last_atr=0;
   last_body=0;
//--- установим правильный объем
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- создадим хендл индикатора ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: не удалось создать iATR, код ошибки %d",__FUNCTION__,GetLastEr
      return(INIT_FAILED);
     }
//--- успешная инициализация эксперта
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- торговый сигнал
   static int signal=0; // +1 означает сигнал на покупку, -1 означает сигнал на продаж
//--- проверим и закроем старые позиции, открытые более holdbars баров назад
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- проверим появление нового бара
   if(isNewBar())
     {
      //--- проверим наличие сигнала
      signal=CheckSignal();
     }
//--- если открыта неттинговая позиция, то сигнал пропускаем - ждем, пока она закроет
```

© 2000-2025, MetaQuotes Ltd.

```
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // выходим из обработчика события NewTick и не входим в рынок до появле
     }
//--- для хеджингового счета каждая позиция живет и закрывается раздельно
   if(signal!=0)
     {
      //--- сигнал на покупку
      if(signal>0)
        {
         PrintFormat("%s: Есть сигнал на покупку! Revers=%s",__FUNCTION__,string(reve
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- сигнал на продажу
      if(signal<0)
        {
         PrintFormat("%s: Есть сигнал на продажу! Revers=%s",__FUNCTION__,string(reve
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- конец функции OnTick
  }
//+------------------------------------------------------------------+
//| Проверяет наличие торгового сигнала                              |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 означает отсутствие сигнала
   int res=0;
//--- получим значение ATR на предпоследнем завершенном баре (индекс бара равен 2)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- получим данные последнего закрытого бара в массив типа MqlRates
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- вычислим размер тела бара на последнем закрытом баре
         last_body=bar[0].close-bar[0].open;
         //--- если тело последнего бара (с индексом 1) превышает предыдущее значение
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // для растущей свечи положительное значение
        }
      else
         PrintFormat("%s: Не удалось получить последний бар! Ошибка",__FUNCTION__,GetI
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    else
       PrintFormat("%s: Не удалось получить значение индикатора ATR! Ошибка",__FUNCTION
//--- если включен реверсивный режим торговли
    res=revers?-res:res;  // если нужно, то развернем сигнал (вместо 1 вернем -1, а вме
//--- вернем значение торгового сигнала
    return (res);
  }
//+------------------------------------------------------------------+
//|  Возвращает true при появлении нового бара                       |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // храним время открытия текущего бара
//--- получим время открытия нулевого бара
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- если время открытия изменилось, значит появился новый бар
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- нужно ли выводить в лог информацию о времени открытия нового бара
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- выведем сообщение о времени открытия нового бара
         PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- получим данные о последнем тике
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() failed, error = ",GetLastError());
         //--- выведем время последнего тика с точностью до миллисекунд
         PrintFormat("Last tick was at %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%100(
        }
      //--- у нас есть новый бар
      return (true);
     }
//--- нового бара нет
   return (false);
  }
//+------------------------------------------------------------------+
//| Покупка по рынку с заданным объемом                              |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- покупаем по рыночной цене
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Продажа по рынку с заданным объемом                              |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- продаем по рыночной цене
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Закрытие позиций по времени удержания в барах                    |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // количество открытых позиций
//--- перебор всех открытых позиций
   for(int i=total-1; i>=0; i--)
     {
      //--- параметры позиции
      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;

      //--- если позиция живет уже долго, а также MagicNumber и символ совпадают
      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); // понижаем регистр текста для правильного форматиро
         PrintFormat("Закрываем позицию #%I64u %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- установка типа ордера и отправки торгового запроса
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Подготовка и отправка торгового запроса                          |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
  {
//--- объявление и инициализация структур
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- параметры запроса
   request.action   =TRADE_ACTION_DEAL;                     // тип торговой операции
   request.position =pos_ticket;                            // тикет позиции, если зан
   request.symbol   =Symbol();                              // символ
   request.volume   =volume;                                // объем
   request.type     =type;                                  // тип ордера
   request.price    =price;                                 // цена совершения сделки
   request.deviation=slip;                                  // допустимое отклонение
   request.magic    =magicnumber;                           // MagicNumber ордера
//--- отправка запроса
   if(!OrderSend(request,result))
     {
      //--- выведем информацию о неудаче
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- сообщим об успешной операции
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Смотри также

Тестирование торговых стратегий, Работа с результатами оптимизации, OnTesterDeinit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2025, MetaQuotes Ltd.

OnTesterDeinit

Вызывается в экспертах при наступлении события TesterDeinit для выполнения необходимых действий по окончании оптимизации эксперта.

```
void  OnTesterDeinit(void);
```

Возвращаемое значение

Нет возвращаемого значения

Примечание

Событие TesterDeinit автоматически генерируется по окончании оптимизации эксперта в тестере стратегий.

Эксперт, имеющий обработчик OnTesterDeinit() или OnTesterPass(), при запуске оптимизации автоматически загружается на отдельном графике терминала с указанными в тестере символом и периодом. Функция предназначена для финальной обработки всех результатов оптимизации.

Необходимо помнить, что фреймы оптимизации, отсылаемые агентами тестирования с помощью функции FrameAdd(), могут приходить пачками и для их доставки требуется время. Поэтому не все фреймы, а соответственно и события TesterPass, могут до окончания оптимизации поступить и быть обработанными в OnTesterPass(). Поэтому для гарантированного получения всех запоздавших фреймов в OnTesterDeinit() необходимо поместить блок кода с использованием функции FrameNext().

Смотри также

Тестирование торговых стратегий, Работа с результатами оптимизации, TesterStatistics, OnTesterInit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2025, MetaQuotes Ltd.

OnTesterPass

Вызывается в экспертах при наступлении события TesterPass для обработки нового фрейма данных во время оптимизации эксперта.

```
void  OnTesterPass(void);
```

Возвращаемое значение

Нет возвращаемого значения

Примечание

Событие TesterPass автоматически генерируется при поступлении фрейма во время оптимизации эксперта в тестере стратегий.

Эксперт, имеющий обработчик OnTesterDeinit() или OnTesterPass(), при запуске оптимизации автоматически загружается на отдельном графике терминала с указанными в тестере символом и периодом. Функция предназначена для обработки фреймов, полученных от агентов тестирования во время оптимизации. Отправку фрейма с результатами тестирования необходимо выполнять из обработчика OnTester() с помощью функции FrameAdd().

Необходимо помнить, что фреймы оптимизации, отсылаемые агентами тестирования с помощью функции FrameAdd(), могут приходить пачками и для их доставки требуется время. Поэтому не все фреймы, а соответственно и события TesterPass, могут до окончания оптимизации поступить и быть обработанными в OnTesterPass(). Поэтому для гарантированного получения всех запоздавших фреймов в OnTesterDeinit() необходимо поместить блок кода с использованием функции FrameNext().

После завершения оптимизации OnTesterDeinit() можно заново перебрать все полученные фреймы с помощью функций FrameFirst()/FrameFilter и FrameNext().

Смотри также

Тестирование торговых стратегий, Работа с результатами оптимизации, OnTesterInit, OnTesterDeinit, FrameFirst, FrameFilter, FrameNext, FrameInputs

© 2000-2025, MetaQuotes Ltd.

Получение рыночной информации

Функции для получения информации о состоянии рынка.

Функция	Действие
SymbolsTotal	Возвращает количество доступных (выбранных в MarketWatch или всех) символов
SymbolExist	Проверяет наличие символа с указанным именем
SymbolName	Возвращает наименование указанного символа
SymbolSelect	Выбирает символ в окне MarketWatch или убирает символ из окна
SymbolIsSynchronized	Проверяет синхронизированность данных по указанному символу в терминале с данными на торговом сервере
SymbolInfoDouble	Возвращает значение типа double указанного символа для соответствующего свойства
SymbolInfoInteger	Возвращает значение целочисленного типа (long, datetime, int или bool) указанного символа для соответствующего свойства
SymbolInfoString	Возвращает значение типа string указанного символа для соответствующего свойства
SymbolInfoMarginRate	Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера
SymbolInfoTick	Возвращает текущие цены для указанного символа в переменной типа MqlTick
SymbolInfoSessionQuote	Позволяет получить время начала и время окончания указанной котировочной сессии для указанных символа и дня недели.
SymbolInfoSessionTrade	Позволяет получить время начала и время окончания указанной торговой сессии для указанных символа и дня недели.
MarketBookAdd	Обеспечивает открытие стакана цен по указанному инструменту, а также производит подписку на получение извещений об изменении указанного стакана
MarketBookRelease	Обеспечивает закрытие стакана цен по указанному инструменту, а также отменяет подписку на получение извещений об изменении указанного стакана
MarketBookGet	Возвращает массив структур типа MqlBookInfo, содержащий записи стакана цен указанного символа

© 2000-2025, MetaQuotes Ltd.

SymbolsTotal

Возвращает количество доступных (выбранных в MarketWatch или всех) символов.

```
int  SymbolsTotal(
   bool  selected      // true - только символы в MarketWatch
   );
```

Параметры

selected

[in] Режим запроса. Может принимать значения true или false.

Возвращаемое значение

Если параметр selected равно true, то возвращается количество выбранных в MarketWatch символов. Если значение false, то возвращается общее количество всех символов.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим количество доступных символов
   int total    = SymbolsTotal(false);       // все символы на сервере
   int selected = SymbolsTotal(true);        // все символы, добавленные в окно "Обзор

//--- выведем полученные данные в журнал
   PrintFormat("Number of available symbols on the server: %d\n"+
               "Number of available symbols selected in MarketWatch: %d", total, selec
   /*
   результат:
   Number of available symbols on the server: 140
   Number of available symbols selected in MarketWatch: 11
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolExist

Проверяет наличие символа с указанным именем.

```
bool  SymbolExist(
   const string  name,     // имя символа
   bool&   is_custom       // признак пользовательского символа
   );
```

Параметры

name

[in] Имя символа.

is_custom

[out] Признак пользовательского символа, который выставляется при успешном выполнении. Если значение равно true, то найденный символ является пользовательским.

Возвращаемое значение

Возвращает false, если символ не найден ни среди стандартных, ни среди пользовательских символов.

Пример:

```
#define SYMBOL_NAME "GBPUSDn"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявляем флаг пользовательского символа и проверяем наличие символа с указанным
   bool custom = false;
   bool result = SymbolExist(SYMBOL_NAME, custom);

//--- объявляем текст сообщения по умолчанию о ненайденном символе
   string text = StringFormat("The symbol '%s' was not found among either the standard

//--- если символ найден - создадим текст сообщения в зависимости от того, в каком спи
   if(result)
     {
      //--- если это стандартный символ
      if(!custom)
         text = StringFormat("The '%s' symbol is available on the server.", SYMBOL_NAM
      //--- если это пользовательский символ
      else
         text = StringFormat("The symbol '%s' was found in the list of custom symbols.
     }

//--- выводим сообщение о результате проверки в журнал
   Print(text);
```

© 2000-2025, MetaQuotes Ltd.

```
   /*
   результат для стандартного символа 'GBPUSD':
   The 'GBPUSD' symbol is available on the server.

   результат для пользовательского символа 'GBPUSDx':
   The symbol 'GBPUSDx' was found in the list of custom symbols.

   результат для отсутствующего символа 'GBPUSDn':
   The symbol 'GBPUSDn' was not found among either the standard or custom symbols.
   */
  }
```

Смотри также

SymbolsTotal, SymbolSelect, Пользовательские символы

© 2000-2025, MetaQuotes Ltd.

SymbolName

Возвращает наименование указанного символа.

```
string  SymbolName(
   int   pos,           // номер в списке
   bool  selected       // true - только символы в MarketWatch
   );
```

Параметры

pos

[in] Номер символа по порядку.

selected

[in] Режим запроса. Если значение true, то символ берется из списка выбранных в MarketWatch. Если значение false, то символ берется из общего списка.

Возвращаемое значение

Значение типа string с именем символа.

Пример:

```
#define SYMBOL_NAME "GBPHKD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- зададим флаг результата поиска символа на сервере
   bool found = false;

//--- найдём символ 'SYMBOL_NAME' в списке всех символов сервера
   int total = SymbolsTotal(false);
   for(int i=0; i<total; i++)
     {
      //--- получаем имя символа в списке по индексу цикла
      string name = SymbolName(i, false);

      //--- если это искомый символ - распечатаем в журнале его имя и позицию в списке
      if(name == SYMBOL_NAME)
        {
         PrintFormat("The '%s' symbol was found in the list of server symbols at index
         found = true;
         break;
        }
     }

//--- если символ не найден на сервере - сообщим об этом перед завершением работы
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!found)
      PrintFormat("The '%s' symbol was not found on the server.", SYMBOL_NAME);
}
```

© 2000-2025, MetaQuotes Ltd.

SymbolSelect

Выбирает символ в окне MarketWatch (Обзор рынка) или убирает символ из этого окна.

```
bool  SymbolSelect(
   string  name,        // имя символа
   bool    select       // включить или выключить
   );
```

Параметры

name

[in] Имя символа.

select

[in] Переключатель. Если значение false, то символ должен быть убран из окна MarketWatch, в противном случае символ должен быть выбран в окно MarketWatch. Символ не может быть убран, если есть открытые графики с этим символом или есть открытые позиции по этому символу.

Возвращаемое значение

В случае неудачи функция возвращает false.

Пример:

```
#define SYMBOL_NAME "GBPHKD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- проверим наличие символа в списках, если нет - сообщаем об этом и завершаем рабо
   bool custom = false;
   if(!SymbolExist(SYMBOL_NAME, custom))
     {
      PrintFormat("'%s' symbol not found in the lists", SYMBOL_NAME);
      return;
     }

//--- добавим символ в окно "Обзор рынка"
   ResetLastError();
   if(!SymbolSelect(SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }
//--- если символ успешно добавлен в список - получим его индекс в окне "Обзор рынка"
   int index = SymbolIndex(SYMBOL_NAME);
   PrintFormat("The '%s' symbol has been added to the MarketWatch list. Symbol index i
```

© 2000-2025, MetaQuotes Ltd.

```
//--- теперь удалим символ из окна "Обзор рынка"
   ResetLastError();
   if(!SymbolSelect(SYMBOL_NAME, false))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }
//--- если символ успешно убран из списка - его индекс в окне "Обзор рынка" равен -1,
   index = SymbolIndex(SYMBOL_NAME);
   PrintFormat("The '%s' symbol has been removed from the MarketWatch list. Symbol inc

   /*
   результат:
   The 'GBPHKD' symbol has been added to the MarketWatch list. Symbol index in the lis
   The 'GBPHKD' symbol has been removed from the MarketWatch list. Symbol index in the
   */
  }
//+------------------------------------------------------------------+
//| Возвращает индекс символа в списке символов окна "Обзор рынка"   |
//+------------------------------------------------------------------+
int SymbolIndex(const string symbol)
  {
   int total = SymbolsTotal(true);
   for(int i=0; i<total; i++)
     {
      string name = SymbolName(i, true);
      if(name == symbol)
         return i;
     }
   return(WRONG_VALUE);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolIsSynchronized

Проверяет факт синхронизированности данных по указанному символу в терминале с данными на торговом сервере

```
bool  SymbolIsSynchronized(
   string  name,        // имя символа
   );
```

Параметры

name

[in] Имя символа.

Возвращаемое значение

Если данные синхронизированы, возвращает true, иначе возвращает false.

Пример:

```
#define SYMBOL_NAME "EURUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем флаг синхронизированности данных в терминале по символу 'SYMBOL_NAME' с
   bool    sync = SymbolIsSynchronized(SYMBOL_NAME);

//--- создаём сообщение в зависимости от флага синхронизированности
   string text = StringFormat("The data on the '%s' symbol in the terminal is synchron
   if(!sync)
      text = StringFormat("The data for the '%s' symbol in the terminal is not synchr

//--- выводим полученный результат в журнал
   Print(text);

   /*
   результат для синхронизированных данных:
   The data on the 'EURUSD' symbol in the terminal is synchronized with the data on th

   результат для ещё не синхронизированных данных:
   The data for the 'GBPHKD' symbol in the terminal is not synchronized with the data
   */
  }
```

Смотри также

SymbolInfoInteger, Организация доступа к данным

© 2000-2025, MetaQuotes Ltd.

SymbolInfoDouble

Возвращает соответствующее свойство указанного символа. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
double  SymbolInfoDouble(
   string                   name,          // символ
   ENUM_SYMBOL_INFO_DOUBLE  prop_id        // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  SymbolInfoDouble(
   string                   name,          // символ
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,       // идентификатор свойства
   double&                  double_var     // сюда примем значение свойства
   );
```

Параметры

name

[in] Имя символа.

prop_id

[in] Идентификатор свойства символа. Значение может быть одним из значений перечисления ENUM_SYMBOL_INFO_DOUBLE.

double_var

[out] Переменная типа double, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа double. В случае неудачного выполнения информацию об ошибке можно получить с помощью функции GetLastError():

- 5040 - неверный строковый параметр для указания имени символа,
- 4301 - неизвестный символ (финансовый инструмент),
- 4302 - символ не выбран в "Обзоре рынка" (нет в списке доступных),
- 4303 - неверный идентификатор свойства символа.

Примечание

Если функция используется для получения информации о последнем тике, то лучше использовать SymbolInfoTick(). Вполне возможно, что по данному символу с момента подключения терминала к торговому счету не было еще ни одной котировки. В таком случае запрашиваемое значение будет неопределенным.

В большинстве случаев достаточно использовать функцию SymbolInfoTick(), которая позволяет получить за один вызов значения Ask, Bid, Last, Volume и время прихода последнего тика.

Для получения информации о размере взимаемой маржи в зависимости от типа и направления ордера используется функция SymbolInfoMarginRate().

© 2000-2025, MetaQuotes Ltd.

Пример:

```
void OnTick()
  {
//--- получим спред из свойств символа
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Спред %s = %I64d пунктов\r\n",
                            spreadfloat?"плавающий":"фиксированный",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- вычислим теперь спред сами
   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+"Вычисленный спред = "+(string)spread_points+" пунктов";
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoInteger

Возвращает соответствующее свойство указанного символа. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
long  SymbolInfoInteger(
   string                     name,      // символ
   ENUM_SYMBOL_INFO_INTEGER   prop_id    // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  SymbolInfoInteger(
   string                     name,      // символ
   ENUM_SYMBOL_INFO_INTEGER   prop_id,   // идентификатор свойства
   long&                      long_var   // сюда примем значение свойства
   );
```

Параметры

name

[in] Имя символа.

prop_id

[in] Идентификатор свойства символа. Значение может быть одним из значений перечисления ENUM_SYMBOL_INFO_INTEGER.

long_var

[out] Переменная типа long, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа long. В случае неудачного выполнения информацию об ошибке можно получить с помощью функции GetLastError():

- 5040 - неверный строковый параметр для указания имени символа,
- 4301 - неизвестный символ (финансовый инструмент),
- 4302 - символ не выбран в "Обзоре рынка" (нет в списке доступных),
- 4303 - неверный идентификатор свойства символа.

Примечание

Если функция используется для получения информации о последнем тике, то лучше использовать SymbolInfoTick(). Вполне возможно, что по данному символу с момента подключения терминала к торговому счету не было еще ни одной котировки. В таком случае запрашиваемое значение будет неопределенным.

В большинстве случаев достаточно использовать функцию SymbolInfoTick(), которая позволяет получить за один вызов значения Ask, Bid, Last, Volume и время прихода последнего тика.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
void OnTick()
  {
//--- получим спред из свойств символа
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Спред %s = %I64d пунктов\r\n",
                            spreadfloat?"плавающий":"фиксированный",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- вычислим теперь спред сами
   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+"Вычисленный спред = "+(string)spread_points+" пунктов";
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoString

Возвращает соответствующее свойство указанного символа. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
string  SymbolInfoString(
   string  name,                        // символ
   ENUM_SYMBOL_INFO_STRING  prop_id     // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  SymbolInfoString(
   string  name,                        // символ
   ENUM_SYMBOL_INFO_STRING  prop_id,    // идентификатор свойства
   string& string_var                   // сюда примем значение свойства
   );
```

Параметры

name

[in] Имя символа.

prop_id

[in] Идентификатор свойства символа. Значение может быть одним из значений перечисления ENUM_SYMBOL_INFO_STRING.

string_var

[out] Переменная типа string, принимающая значение запрашиваемого свойства. В случае неудачного выполнения информацию об ошибке можно получить с помощью функции GetLastError():

- 5040 - неверный строковый параметр для указания имени символа,
- 4301 - неизвестный символ (финансовый инструмент),
- 4302 - символ не выбран в "Обзоре рынка" (нет в списке доступных),
- 4303 - неверный идентификатор свойства символа.

Примечание

Если функция используется для получения информации о последнем тике, то лучше использовать SymbolInfoTick(). Вполне возможно, что по данному символу с момента подключения терминала к торговому счету не было еще ни одной котировки. В таком случае запрашиваемое значение будет неопределенным.

В большинстве случаев достаточно использовать функцию SymbolInfoTick(), которая позволяет получить за один вызов значения Ask, Bid, Last, Volume и время прихода последнего тика.

Пример:

```
#define SYMBOL_NAME "USDJPY"
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим строковые данные по символу SYMBOL_NAME
   string currency_base   = SymbolInfoString(SYMBOL_NAME, SYMBOL_CURRENCY_BASE);     //
   string currency_profit = SymbolInfoString(SYMBOL_NAME, SYMBOL_CURRENCY_PROFIT);   //
   string currency_margin = SymbolInfoString(SYMBOL_NAME, SYMBOL_CURRENCY_MARGIN);   //
   string symbol_descript = SymbolInfoString(SYMBOL_NAME, SYMBOL_DESCRIPTION);       //

//--- создадим текст сообщения с полученными данными
   string text=StringFormat("Symbol %s:\n"+
                            "- Currency Base: %s\n"+
                            "- Currensy Profit: %s\n"+
                            "- Currency Margin: %s\n"+
                            "- Symbol Description: %s",
                            SYMBOL_NAME, currency_base,
                            currency_profit, currency_margin,
                            symbol_descript);

//--- выведем полученные данные в журнал
   Print(text);
   /*
   Symbol USDJPY:
   - Currency Base: USD
   - Currensy Profit: JPY
   - Currency Margin: USD
   - Symbol Description: US Dollar vs Yen
   */
  }
```

Возвращаемое значение

Значение типа string.

© 2000-2025, MetaQuotes Ltd.

SymbolInfoMarginRate

Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера.

```
bool  SymbolInfoMarginRate(
   string              name,                    // символ
   ENUM_ORDER_TYPE     order_type,              // тип ордера
   double&             initial_margin_rate,     // коэффициент взимания начальной марж
   double&             maintenance_margin_rate  // коэффициент взимания поддерживающей
   );
```

Параметры

name

[in] Имя символа.

order_type

[in] Тип ордера.

initial_margin_rate

[in] Переменная типа double для получения коэффициента взимания начальной маржи. Начальная маржа - это размер гарантийной суммы под совершение сделки объемом в 1 лот соответствующего направления. Умножая коэффициент на начальную маржу, мы можем получить размер средств, который будет зарезервирован на счете при размещении ордера указанного типа.

maintenance_margin_rate

[out] Переменная типа double для получения коэффициента взимания поддерживающей маржи. Поддерживающая маржа - это размер минимальной суммы для поддержания открытой позиции объемом в 1 лот соответствующего направления. Умножая коэффициент на поддерживающую маржу, мы можем получить размер средств, который будет зарезервирован на счете после срабатывания ордера указанного типа.

Возвращаемое значение

Возвращает true в случае удачного выполнения запроса свойств, иначе false.

Пример:

```
#define SYMBOL_NAME Symbol()

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим переменные, в которые будут записаны коэффициенты
   double initial_margin_rate = 0;      // коэффициент взимания начальной маржи
   double maintenance_margin_rate = 0; // коэффициент взимания поддерживающей маржи

//--- получим и выведем в журнал коэффициенты по символу SYMBOL_NAME для рыночного ор
   SymbolInfoMarginRatePrint(SYMBOL_NAME, ORDER_TYPE_BUY, initial_margin_rate, mainter
```

© 2000-2025, MetaQuotes Ltd.

```
//--- получим и выведем в журнал коэффициенты по символу SYMBOL_NAME для рыночного орд
   SymbolInfoMarginRatePrint(SYMBOL_NAME, ORDER_TYPE_SELL, initial_margin_rate, mainte

   /*
   результат:
   Initial margin rate for symbol AFKS for the Buy market order: 0.225600
   Maintenance margin rate for symbol AFKS for the Buy market order: 0.112800
   Initial margin rate for symbol AFKS for the Sell market order: 0.254400
   Maintenance margin rate for symbol AFKS for the Sell market order: 0.127200
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал коэффициенты взимания маржи                     |
//+------------------------------------------------------------------+
void SymbolInfoMarginRatePrint(const string symbol, const ENUM_ORDER_TYPE order_type,
  {
//--- получим коэффициенты по символу symbol для рыночного ордера order_type
   ResetLastError();
   if(!SymbolInfoMarginRate(symbol, order_type, initial_margin_rate, maintenance_margi
     {
      Print("SymbolInfoMarginRate() failed. Error ", GetLastError());
      return;
     }
//--- распечатаем полученные значения коэффициентов
   string type=(order_type==ORDER_TYPE_BUY ? "Buy" : order_type==ORDER_TYPE_SELL ? "Se
   PrintFormat("Initial margin rate for symbol %s for the %s market order: %f\n"+
               "Maintenance margin rate for symbol %s for the %s market order: %f",
               SYMBOL_NAME, type, initial_margin_rate,
               SYMBOL_NAME, type, maintenance_margin_rate);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoTick

Возвращает текущие цены для указанного символа в переменной типа MqlTick.

```
bool  SymbolInfoTick(
   string    symbol,      // символ
   MqlTick&  tick         // ссылка на структуру
   );
```

Параметры

symbol

[in] Имя символа.

tick

[out] Ссылка на структуру типа MqlTick, в которую будут помещены текущие цены и время последнего обновления цен.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
#define SYMBOL_NAME "EURUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим массив с типом структуры MqlTick размерностью 1
   MqlTick tick[1]={};

//--- получим последние цены по символу SYMBOL_NAME в структуру MqlTick
   if(!SymbolInfoTick(SYMBOL_NAME, tick[0]))
     {
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return;
     }

//--- выведем полученные данные в журнал
   PrintFormat("Latest price data for the '%s' symbol:", SYMBOL_NAME);
   ArrayPrint(tick);
   /*
   результат:
   Latest price data for the 'EURUSD' symbol:
                    [time]    [bid]    [ask] [last] [volume]     [time_msc] [flags] [volu
   [0] 2024.05.17 23:58:54 1.08685 1.08695 0.0000        0 1715990334319       6
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoSessionQuote

Позволяет получить время начала и время окончания указанной котировочной сессии для указанных символа и дня недели.

```
bool  SymbolInfoSessionQuote(
   string            name,                // имя символа
   ENUM_DAY_OF_WEEK  day_of_week,         // день недели
   uint              session_index,       // номер сессии
   datetime&         from,                // время начала сессии
   datetime&         to                   // время окончания сессии
   );
```

Параметры

name

[in] Имя символа.

ENUM_DAY_OF_WEEK

[in] День недели, значение из перечисления ENUM_DAY_OF_WEEK.

uint

[in] Порядковый номер сессии, для которой нужно получить время начала и время окончания. Индексация сессий начинается с 0.

from

[out] Время начала сессии в секундах от 00 часов 00 минут, в полученном значении дату следует игнорировать.

to

[out] Время окончания сессии в секундах от 00 часов 00 минут, в полученном значении дату следует игнорировать.

Возвращаемое значение

Если данные для указанных сессии, символа и дня недели получены, то возвращает true, иначе возвращает false.

Пример:

```
#define SYMBOL_NAME   Symbol()
#define SESSION_INDEX 0

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- распечатаем заголовок с символом и индексом сессии SESSION_INDEX и
//--- в цикле по дням недели с ПН по ПТ распечатаем в журнале время начала и конца ко
   PrintFormat("Symbol %s, Quote session %d:", SYMBOL_NAME, SESSION_INDEX);
   for(int i=MONDAY; i<SATURDAY; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      SymbolInfoSessionQuotePrint(SYMBOL_NAME, (ENUM_DAY_OF_WEEK)i, SESSION_INDEX);
   /*
   результат:
   Symbol RU000A103661, Quote session 0:
   - Monday     06:45 - 00:00
   - Tuesday    06:45 - 00:00
   - Wednesday  06:45 - 00:00
   - Thursday   06:45 - 00:00
   - Friday     06:45 - 00:00
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал время начала и окончания указанной              |
//| котировочной сессии для указанных символа и дня недели           |
//+------------------------------------------------------------------+
void SymbolInfoSessionQuotePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- объявим переменные для записи начала и конца котировочной сессии
   datetime date_from;  // время начала сессии
   datetime date_to;    // время окончания сессии

//--- получаем данные котировочной сессии по символу и дню недели
   if(!SymbolInfoSessionQuote(symbol, day_of_week, session_index, date_from, date_to))
     {
      Print("SymbolInfoSessionQuote() failed. Error ", GetLastError());
      return;
     }

//--- создаём наименование дня недели из константы перечисления
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- выводим в журнал данные по указанной котировочной сессии
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

Смотри также

Информация об инструменте, TimeToStruct, Структура даты

© 2000-2025, MetaQuotes Ltd.

SymbolInfoSessionTrade

Позволяет получить время начала и время окончания указанной торговой сессии для указанных символа и дня недели.

```
bool  SymbolInfoSessionTrade(
   string            name,                // имя символа
   ENUM_DAY_OF_WEEK  day_of_week,         // день недели
   uint              session_index,       // номер сессии
   datetime&         from,                // время начала сессии
   datetime&         to                   // время окончания сессии
   );
```

Параметры

name

[in] Имя символа.

ENUM_DAY_OF_WEEK

[in] День недели, значение из перечисления ENUM_DAY_OF_WEEK.

uint

[in] Порядковый номер сессии, для которой нужно получить время начала и время окончания. Индексация сессий начинается с 0.

from

[out] Время начала сессии в секундах от 00 часов 00 минут, в полученном значении дату следует игнорировать.

to

[out] Время окончания сессии в секундах от 00 часов 00 минут, в полученном значении дату следует игнорировать.

Возвращаемое значение

Если данные для указанных сессии, символа и дня недели получены, то возвращает true, иначе возвращает false.

Пример:

```
#define SYMBOL_NAME   Symbol()
#define SESSION_INDEX 0

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- распечатаем заголовок с символом и индексом сессии SESSION_INDEX и
//--- в цикле по дням недели с ПН по ПТ распечатаем в журнале время начала и конца тор
   PrintFormat("Symbol %s, Trade session %d:", SYMBOL_NAME, SESSION_INDEX);
   for(int i=MONDAY; i<SATURDAY; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      SymbolInfoSessionTradePrint(SYMBOL_NAME, (ENUM_DAY_OF_WEEK)i, SESSION_INDEX);
   /*
   результат:
   Symbol GBPUSD, Trade session 0:
   - Monday     00:15 - 23:55
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал время начала и окончания указанной              |
//| торговой сессии для указанных символа и дня недели               |
//+------------------------------------------------------------------+
void SymbolInfoSessionTradePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- объявим переменные для записи начала и конца торговой сессии
   datetime date_from;  // время начала сессии
   datetime date_to;    // время окончания сессии

//--- получаем данные торговой сессии по символу и дню недели
   if(!SymbolInfoSessionTrade(symbol, day_of_week, session_index, date_from, date_to))
     {
      Print("SymbolInfoSessionTrade() failed. Error ", GetLastError());
      return;
     }

//--- создаём наименование дня недели из константы перечисления
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- выводим в журнал данные по указанной торговой сессии
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

Смотри также

Информация об инструменте, TimeToStruct, Структура даты

© 2000-2025, MetaQuotes Ltd.

MarketBookAdd

Обеспечивает открытие стакана цен по указанному инструменту, а также производит подписку на получение извещений об изменении указанного стакана.

```
bool  MarketBookAdd(
   string  symbol      // символ
   );
```

Параметры

symbol

[in] Имя символа, чей стакан цен предполагается использовать в данном эксперте или скрипте.

Возвращаемое значение

Значение true в случае успешного открытия, иначе false.

Примечание

Обычно, эта функция должна вызываться из функции OnInit() или в конструкторе класса. Для обработки приходящих извещений в программе эксперта должна присутствовать функция void OnBookEvent(string& symbol).

Пример:

```
#define   SYMBOL_NAME   "GBPUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- откроем стакан цен по символу SYMBOL_NAME
   if(!MarketBookAdd(SYMBOL_NAME))
     {
      PrintFormat("MarketBookAdd(%s) failed. Error ", SYMBOL_NAME, GetLastError());
      return;
     }

//--- выведем в журнал сообщение об успешном открытии стакана цен
   PrintFormat("The MarketBook for the '%s' symbol was successfully opened and a subsc

//--- подождём 2 секунды
   Sleep(2000);

//--- по окончании отпишемся от открытого стакана цен
   ResetLastError();
   if(MarketBookRelease(SYMBOL_NAME))
      PrintFormat("MarketBook for the '%s' symbol was successfully closed", SYMBOL_NAM
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Error %d occurred when closing MarketBook using the '%s' symbol", G

   /*
   результат:
   The MarketBook for the 'GBPUSD' symbol was successfully opened and a subscription t
   MarketBook for the 'GBPUSD' symbol was successfully closed
   */
  }
```

Смотри также

Структура стакана цен, Структуры и классы

© 2000-2025, MetaQuotes Ltd.

MarketBookRelease

Обеспечивает закрытие стакана цен по указанному инструменту, а также отменяет подписку на получение извещений об изменении указанного стакана.

```
bool  MarketBookRelease(
   string  symbol      // имя символа
   );
```

Параметры

symbol

[in] Имя символа.

Возвращаемое значение

Значение true в случае успешного закрытия, иначе false.

Примечание

Обычно эта функция должна вызываться из функции OnDeinit() в том случае, если в функции OnInit() была вызвана соответствующая функция MarketBookAdd(). Либо должна вызываться из деструктора класса, если в конструкторе этого класса вызывается соответствующая функция MarketBookAdd().

Пример:

```
#define SYMBOL_NAME "GBPUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- откроем стакан цен по символу SYMBOL_NAME
   if(!MarketBookAdd(SYMBOL_NAME))
     {
      PrintFormat("MarketBookAdd(%s) failed. Error ", SYMBOL_NAME, GetLastError());
      return;
     }

//--- выведем в журнал сообщение об успешном открытии стакана цен
   PrintFormat("The MarketBook for the '%s' symbol was successfully opened and a subsc

//--- подождём 2 секунды
   Sleep(2000);

//--- по завершении работы выполним отписку от открытого стакана цен
//--- в журнал выведем сообщения об успешной отписке от стакана цен, либо об ошибке
   ResetLastError();
   if(MarketBookRelease(SYMBOL_NAME))
      PrintFormat("MarketBook for the '%s' symbol was successfully closed", SYMBOL_NAM
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      PrintFormat("Error %d occurred when closing MarketBook using the '%s' symbol", G

   /*
   результат:
   The MarketBook for the 'GBPUSD' symbol was successfully opened and a subscription t
   MarketBook for the 'GBPUSD' symbol was successfully closed
   */
  }
```

Смотри также

Структура стакана цен, Структуры и классы

© 2000-2025, MetaQuotes Ltd.

MarketBookGet

Возвращает массив структур MqlBookInfo, содержащий записи стакана цен указанного символа.

```
bool  MarketBookGet(
   string        symbol,      // символ
   MqlBookInfo&  book[]       // ссылка на массив
   );
```

Параметры

symbol

[in] Имя символа.

book[]

[out] Ссылка на массив записей стакана цен. Массив может быть заранее распределен для достаточного количества записей. Если динамический массив не был заранее распределен в оперативной памяти, то клиентский терминал сам распределит этот массив.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Стакан цен должен быть предварительно открыт функцией MarketBookAdd().

Пример:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo по ",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+":",priceArray[i].price
               +"    Volume= "+priceArray[i].volume,
               " type = ",priceArray[i].type);
        }
     }
   else
     {
      Print("Не удалось получить содержимое стакана по символу ",Symbol());
     }
```

Смотри также

Структура стакана цен, Структуры и классы

© 2000-2025, MetaQuotes Ltd.

Функциии Экономического Календаря

В этом разделе описываются функции для работы с Экономическим календарем, который доступен прямо в платформе MetaTrader. Экономический календарь является готовой энциклопедией с описанием макроэкономических индикаторов, даты их выхода и степени важности. Актуальные значения макроэкономических показателей поступают в платформу MetaTrader моментально прямо в момент публикации и отображаются на графике в виде меток – это позволяет визуально отслеживать нужные показатели в разрезе стран, валют и важности.

Все функции для работы с Экономическим календарем используют время торгового сервера (TimeTradeServer). Это означает, что время в структуре MqlCalendarValue и входящие параметры времени в функциях CalendarValueHistoryByEvent/CalendarValueHistory задаются в таймзоне торгового сервера, а не в локальном времени пользователя.

Функции Экономического календаря позволяют проводить автоматический анализ поступающих событий по собственным критериям важности и в разрезе нужных стран/валют.

Функция	Действие
CalendarCountryById	Получает описание страны по её идентификатору
CalendarEventById	Получает описание события по его идентификатору
CalendarValueById	Получает описание значения события по его идентификатору
CalendarCountries	Получает массив описаний стран, доступных в Календаре
CalendarEventByCountry	Получает массив описаний всех событий, доступных в Календаре, по указанному коду страны
CalendarEventByCurrency	Получает массив описаний всех событий, доступных в Календаре, по указанной валюте
CalendarValueHistoryByEvent	Получает массив значений по всем событиям на заданном диапазоне времени по идентификатору события
CalendarValueHistory	Получает массив значений по всем событиям на заданном диапазоне времени с фильтром по стране и/или валюте
CalendarValueLastByEvent	Получает массив значений события по его ID с момента состояния базы Календаря с заданным change_id
CalendarValueLast	Получает массив значений по всем событиям с фильтрацией по стране и/или валюте с момента состояния базы Календаря с заданным change_id

© 2000-2025, MetaQuotes Ltd.

CalendarCountryById

Получает описание страны по её идентификатору.

```
bool  CalendarCountryById(
   const long           country_id,     // идентификатор страны
   MqlCalendarCountry&  country         // переменная для получения описания страны
   );
```

Параметры

country_id

[in] Идентификатор страны по стандарту ISO 3166-1.

country

[out] Переменная типа MqlCalendarCountry для получения описания страны.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 5402 - ERR_CALENDAR_NO_DATA (страна не найдена),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени).

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим список стран из Экономического календаря
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- проверим результат
   if(count==0)
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
//--- если у нас две и более стран
   if(count>=2)
     {
      MqlCalendarCountry country;
      //--- теперь получим описание страны по её идентификатору
      if(CalendarCountryById(countries[1].id, country))
        {
         //--- подготовим описание страны
         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");
```

© 2000-2025, MetaQuotes Ltd.

```
         descr+=("url_name = " + country.url_name);
         //---выведем описание страны
         Print(descr);
        }
      else
         Print("CalendarCountryById() failed. Error ",GetLastError());
     }
//---
  }
/*
   Результат:
   id = 999
   name = European Union
   code = EU
   currency = EUR
   currency_symbol = €
   url_name = european-union
*/
```

Смотри также

CalendarCountries, CalendarEventByCountry

© 2000-2025, MetaQuotes Ltd.

CalendarEventById

Получает описание события по его идентификатору.

```
bool  CalendarEventById(
   ulong                event_id,      // идентификатор события
   MqlCalendarEvent&    event          // переменная для получения описания события
   );
```

Параметры

event_id

[in] Идентификатор события.

event

[out] Переменная типа MqlCalendarEvent для получения описания события.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 5402 - ERR_CALENDAR_NO_DATA (страна не найдена),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени).

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- код страны для Германии по стандарту ISO 3166-1 Alpha-2
   string germany_code="DE";
//--- получим события для Германии
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(germany_code,events);
//--- выведем события для Германии в Журнал
   if(events_count>0)
     {
      PrintFormat("События для Германии: %d",events_count);
      ArrayPrint(events);
     }
   else
     {
      PrintFormat("Не удалось получить события для кода страны %s, ошибка %d",
                  germany_code,GetLastError());
      //--- досрочное завершение скрипта
      return;
     }
//--- получим описание последнего события из массива events[]
```

© 2000-2025, 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("Получено описания события с event_id=%d",event_id);
      PrintFormat("Страна: %s (код страны = %d)",country.name,event.country_id);
      PrintFormat("Имя события: %s",event.name);
      PrintFormat("Код события: %s",event.event_code);
      PrintFormat("Важность события: %s",EnumToString((ENUM_CALENDAR_EVENT_IMPORTANCE)
      PrintFormat("Тип события: %s",EnumToString((ENUM_CALENDAR_EVENT_TYPE)event.type)
      PrintFormat("Сектор события: %s",EnumToString((ENUM_CALENDAR_EVENT_SECTOR)event.
      PrintFormat("Периодичность события: %s",EnumToString((ENUM_CALENDAR_EVENT_FREQUE
      PrintFormat("Режим выхода события: %s",EnumToString((ENUM_CALENDAR_EVENT_TIMEMOD
      PrintFormat("Единица измерения значения: %s",EnumToString((ENUM_CALENDAR_EVENT_U
      PrintFormat("Количество знаков после запятой: %d",event.digits);
      PrintFormat("Множитель значения: %s",EnumToString((ENUM_CALENDAR_EVENT_MULTIPLIE
      PrintFormat("URL источника: %s",event.source_url);
     }
   else
      PrintFormat("Не удалось получить описание события для event_d=%s, ошибка %d",
                  event_id,GetLastError());
  }
/*
   Результат:
   События для Германии: 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
   Получено описания события с event_id=276500003
   Страна: Germany (код страны = 276)
   Имя события: Markit Composite PMI
   Код события: markit-composite-pmi
   Важность события: CALENDAR_IMPORTANCE_MODERATE
   Тип события: CALENDAR_TYPE_INDICATOR
   Сектор события: CALENDAR_SECTOR_BUSINESS
   Периодичность события: CALENDAR_FREQUENCY_MONTH
   Режим выхода события: CALENDAR_TIMEMODE_DATETIME
   Единица измерения значения: CALENDAR_UNIT_NONE
   Количество знаков после запятой: 1
   Множитель значения: CALENDAR_MULTIPLIER_NONE
   URL источника: https://www.markiteconomics.com
```

© 2000-2025, MetaQuotes Ltd.

```
*/
```

Смотри также

CalendarEventByCountry, CalendarEventByCurrency, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueById

Получает описание значения события по его идентификатору.

```
bool  CalendarValueById(
   ulong                value_id,      // идентификатор значения события
   MqlCalendarValue&    value          // переменная для получения значения события
   );
```

Параметры

value_id

[in] Идентификатор значения события.

value

[out] Переменная типа MqlCalendarValue для получения значения события. Смотри пример обработки событий календаря.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 5402 - ERR_CALENDAR_NO_DATA (страна не найдена),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени).

Примечание

Все функции для работы с Экономическим календарем используют время торгового сервера (TimeTradeServer). Это означает, что время в структуре MqlCalendarValue и входящие параметры времени в функциях CalendarValueHistoryByEvent/CalendarValueHistory задаются в таймзоне торгового сервера, а не в локальном времени пользователя.

Структура MqlCalendarValue предоставляет методы для проверки и получения значений из полей *actual_value*, *forecast_value*, *prev_value* и *revised_prev_value*. Если значение поля не задано, то поле хранит значение LONG_MIN (-9223372036854775808).

При этом необходимо иметь в виду, что значения в этих полях хранятся увеличенными в миллион раз. Это означает, что при получении значений в MqlCalendarValue функциями CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent и CalendarValueLast, значения этих полей нужно проверять на равенство LONG_MIN, и если значение в поле задано, то для получения значения необходимо разделить значение поля на 1 000 000 (миллион). Другой способ получения значений - делать проверку и получать значения функциями самой структуры MqlCalendarValue.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
//--- код страны для Японии по стандарту ISO 3166-1 Alpha-2
   string japan_code="JP";
//--- зададим границы диапазона, из которого берем события
   datetime date_from=D'01.01.2018';  // берем все события с 2018 года
   datetime date_to=0;                // 0 означает все известные события, даже те, чт
//--- получим массив значений событий для Японии
   MqlCalendarValue values[];
   int values_count=CalendarValueHistory(values,date_from,date_to,japan_code);
//--- пройдем по найденным значениям событий
   if(values_count>0)
     {
      PrintFormat("Количество значений для событий Японии: %d",values_count);
      //--- удалим все "пустые" значения (actual_value==-9223372036854775808)
      for(int i=values_count-1;i>=0;i--)
        {
         if(values[i].actual_value==-9223372036854775808)
            ArrayRemove(values,i,1);
        }
      PrintFormat("Количество значений после удаления пустых: %d",ArraySize(values));
     }
   else
     {
      PrintFormat("Не удалось получить события для кода страны %s, ошибка %d",
                  japan_code,GetLastError());
      //--- досрочное завершение скрипта
      return;
     }
//--- оставим не более 10 значений в массиве values[]
   if(ArraySize(values)>10)
     {
      PrintFormat("Уменьшим список значений до 10 и покажем их");
      ArrayRemove(values,0,ArraySize(values)-10);
     }
   ArrayPrint(values);

//--- теперь покажем как на основе известного value_id получить описание значения собы
   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
     }
//---
  }
/*
   Результат:
   Количество значений для событий Японии: 1734
   Количество значений после удаления пустых: 1017
```

© 2000-2025, MetaQuotes Ltd.

```
   Уменьшим список значений до 10 и покажем их
        [id] [event_id]              [time]             [period] [revision] [actual_va
   [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
   Выведем краткую информацию по значениям событий на основе 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
*/
```

Смотри также

CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent, CalendarValueLast

© 2000-2025, MetaQuotes Ltd.

CalendarCountries

Получает массив описаний стран, доступных в Календаре.

```
int  CalendarCountries(
   MqlCalendarCountry&  countries[]         // массив для получения списка описаний ст
   );
```

Параметры

countries[]

[out] Массив типа MqlCalendarCountry для получения описаний всех стран Календаря.

Возвращаемое значение

Количество полученных описаний. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени),
- 5400 - ERR_CALENDAR_MORE_DATA (размер массива недостаточен для получения описаний всех стран, поэтому получили только то, что вместилось).

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим список стран из Экономического календаря
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- выведем массив в Журнал
   if(count>0)
      ArrayPrint(countries);
   else
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
/*
   Результат:
        [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"      "Ŧ"               "japan"
   [ 7]  156 "China"          "CN"   "CNY"      "Ŧ"               "china"
   [ 8]  826 "United Kingdom" "GB"   "GBP"      "J"               "united-kingdom"
   [ 9]  756 "Switzerland"    "CH"   "CHF"      "₣"               "switzerland"
```

© 2000-2025, 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"
*/
  }
```

Смотри также

CalendarEventByCountry, CalendarCountryById

© 2000-2025, MetaQuotes Ltd.

CalendarEventByCountry

Получает массив описаний всех событий, доступных в Календаре, по указанному коду страны.

```
int  CalendarEventByCountry(
   string               country_code,     // кодовое имя страны по ISO 3166-1 alpha-2
   MqlCalendarEvent&    events[]          // переменная для получения массива описаний
   );
```

Параметры

country_code

[in] Кодовое имя страны согласно ISO 3166-1 alpha-2

events[]

[out] Массив типа MqlCalendarEvent для получения описаний всех событий для указанной страны.

Возвращаемое значение

Количество полученных описаний. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 4004 - ERR_NOT_ENOUGH_MEMORY (не достаточно памяти для выполнения запроса),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени),
- ошибки неудачного выполнения ArrayResize()

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- код страны для Европейского союза по стандарту ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- получим события для Евросоюза
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- выведем события для Евросоюза в Журнал
   if(events_count>0)
     {
      PrintFormat("События для Европейского союза: %d",events_count);
      ArrayPrint(events);
     }
//---
  }
/*
   Результат:
   События для Европейского союза: 56
              [id] [type]  [country_id] [unit] [importance] [multiplier] [digits] [ever
```

© 2000-2025, MetaQuotes Ltd.

```
   [ 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
   ...

*/
```

Смотри также

CalendarCountries, CalendarCountryById

© 2000-2025, MetaQuotes Ltd.

CalendarEventByCurrency

Получает массив описаний всех событий, доступных в Календаре, по указанной валюте.

```
int  CalendarEventByCurrency(
   const string         currency,      // кодовое наименование валюты страны
   MqlCalendarEvent&    events[]       // переменная для получения массива описаний
   );
```

Параметры

currency

[in] Кодовое наименование валюты страны.

events[]

[out] Массив типа MqlCalendarEvent для получения описаний всех событий для указанной валюты.

Возвращаемое значение

Количество полученных описаний. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 4004 - ERR_NOT_ENOUGH_MEMORY (не достаточно памяти для выполнения запроса),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени),
- ошибки неудачного выполнения ArrayResize()

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим массив для получения событий Экономического календаря
      MqlCalendarEvent events[];
//--- получим события для валюты Европейского союза
      int count = CalendarEventByCurrency("EUR",events);
      Print("count = ", count);
//--- для примера нам достаточно 10 событий
      if(count>10)
         ArrayResize(events,10);
//--- выведем события в Журнал
      ArrayPrint(events);
  }
/*
   Результат:
              [id] [type] [country_id] [unit] [importance]
   [0] 999010001      0          999      0            2 "https://www.ecb.europa.eu/ho
   [1] 999010002      0          999      0            2 "https://www.ecb.europa.eu/ho
```

© 2000-2025, MetaQuotes Ltd.

```
   [2] 999010003      0          999      0            3 "https://www.ecb.europa.eu/ho
   [3] 999010004      0          999      0            3 "https://www.ecb.europa.eu/ho
   [4] 999010005      0          999      0            2 "https://www.ecb.europa.eu/ho
   [5] 999010006      1          999      1            3 "https://www.ecb.europa.eu/ho
   [6] 999010007      1          999      1            3 "https://www.ecb.europa.eu/ho
   [7] 999010008      0          999      0            2 "https://www.ecb.europa.eu/ho
   [8] 999010009      1          999      2            2 "https://www.ecb.europa.eu/ho
   [9] 999010010      0          999      0            2 "https://www.ecb.europa.eu/ho
*/
```

Смотри также

CalendarEventById, CalendarEventByCountry

© 2000-2025, MetaQuotes Ltd.

CalendarValueHistoryByEvent

Получает массив значений по всем событиям на заданном диапазоне времени по идентификатору события.

```
int  CalendarValueHistoryByEvent(
   ulong              event_id,          // идентификатор события
   MqlCalendarValue&  values[],          // массив для получения описаний значений
   datetime           datetime_from,     // левая граница диапазона времени
   datetime           datetime_to=0      // правая граница диапазона времени
   );
```

Параметры

event_id

[in] Идентификатор события.

values[]

[out] Массив типа MqlCalendarValue для получения значений событиий. Смотри пример обработки событий календаря.

datetime_from

[in] Начальная дата диапазона времени, из которого выбираются события по заданному идентификатору, при этом *datetime_from* < *datetime_to*.

datetime_to=0

[in] Конечная дата диапазона времени, из которого выбираются события по заданному идентификатору. Если параметр *datetime_to* не задан (или равен 0), то будут возвращены все значения событий от заданной даты *datetime_from*, имеющиеся в базе Календаря, в том числе и значения событий, которые запланированы на будущее.

Возвращаемое значение

В случае успеха возвращает количество доступных значений в массиве values, иначе -1. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 4004 - ERR_NOT_ENOUGH_MEMORY (не достаточно памяти для выполнения запроса),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени),
- 5400 - ERR_CALENDAR_MORE_DATA (размер массива недостаточен для получения описаний всех значений, поэтому получили только то, что вместилось),
- ошибки неудачного выполнения ArrayResize()

Если для значения события отсутствует какое либо из полей ниже

```
struct MqlCalendarValue
  {
   ...
   long              actual_value;              // актуальное значение события
   long              prev_value;                // предыдущее значение события
   long              revised_prev_value;        // пересмотренное предыдущее зна
   long              forecast_value;            // прогнозное значение события
```

© 2000-2025, MetaQuotes Ltd.

```
      ...
     };
```

то значение отсутствующего поля вернется как INT64_MIN (-9223372036854775808). Смотри в примере ниже значения поля *revised_prev_value*.

Примечание

Все функции для работы с Экономическим календарем используют время торгового сервера (TimeTradeServer). Это означает, что время в структуре MqlCalendarValue и входящие параметры времени в функциях CalendarValueHistoryByEvent/CalendarValueHistory задаются в таймзоне торгового сервера, а не в локальном времени пользователя.

Структура MqlCalendarValue предоставляет методы для проверки и получения значений из полей *actual_value*, *forecast_value*, *prev_value* и *revised_prev_value*. Если значение поля не задано, то поле хранит значение LONG_MIN (-9223372036854775808).

При этом необходимо иметь в виду, что значения в этих полях хранятся увеличенными в миллион раз. Это означает, что при получении значений в MqlCalendarValue функциями CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent и CalendarValueLast, значения этих полей нужно проверять на равенство LONG_MIN, и если значение в поле задано, то для получения значения необходимо разделить значение поля на 1 000 000 (миллион). Другой способ получения значений - делать проверку и получать значения функциями самой структуры MqlCalendarValue.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- код страны для Европейского союза по стандарту ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- получим события для Евросоюза
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- выведем события для Евросоюза в Журнал
   if(events_count>0)
     {
      PrintFormat("События для Европейского союза: %d",events_count);
      //--- сократим список событий, для изучения достаточно оставить 10
      ArrayResize(events,10);
      ArrayPrint(events);
     }
//--- видими, что событие "ECB Interest Rate Decision"  имеет event_id=999010007
   ulong event_id=events[6].id;         // id этого события может поменяться в Календар
   string event_name=events[6].name;    // имя события из Календаря
   PrintFormat("Получим значения для события event_name=%s event_id=%d",event_name,eve
//--- получим все значения события "ECB Interest Rate Decision"
   MqlCalendarValue values[];
//--- зададим границы диапазона, из которого берем события
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime date_from=0;           // берем все события с начала доступной истории
   datetime date_to=D'01.01.2016'; // берем события не старше 2016 года
   if(CalendarValueHistoryByEvent(event_id,values,date_from,date_to))
     {
      PrintFormat("Получены значения события %s: %d",
                  event_name,ArraySize(values));
      //--- сократим список значений, для изучения достаточно оставить 10
      ArrayResize(values,10);
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Ошибка! Не удалось получить значения для события event_id=%d",event
      PrintFormat("Код ошибки: %d",GetLastError());
     }
  }
//---
/*
   Результат:
   События для Европейского союза: 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
   Получим значения для события event_name=ECB Interest Rate Decision event_id=999010(
   Получены значения события 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(
*/
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

CalendarCountries, CalendarEventByCountry, CalendarValueHistory, CalendarEventById, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueHistory

Получает массив значений по всем событиям на заданном диапазоне времени с фильтром по стране и/или валюте.

```
int  CalendarValueHistory(
   MqlCalendarValue&  values[],              // массив для получения описаний значений
   datetime           datetime_from,         // левая граница диапазона времени
   datetime           datetime_to=0          // правая граница диапазона времени
   const string       country_code=NULL,     // кодовое имя страны по ISO 3166-1 alpha
   const string       currency=NULL          // кодовое наименование валюты страны
   );
```

Параметры

values[]

[out] Массив типа MqlCalendarValue для получения значений событиий. Смотри пример обработки событий календаря.

datetime_from

[in] Начальная дата диапазона времени, из которого выбираются события по заданному идентификатору, при этом *datetime_from* < *datetime_to*.

datetime_to=0

[in] Конечная дата диапазона времени, из которого выбираются события по заданному идентификатору. Если параметр *datetime_to* не задан (или равен 0), то будут возвращены все значения событий от заданной даты *datetime_from*, имеющиеся в базе Календаря, в том числе и значения событий, которые запланированы на будущее.

country_code=NULL

[in] Кодовое имя страны согласно ISO 3166-1 alpha-2

currency=NULL

[in] Кодовое наименование валюты страны.

Возвращаемое значение

В случае успеха возвращает количество доступных значений в массиве values, иначе -1. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 4004 - ERR_NOT_ENOUGH_MEMORY (не достаточно памяти для выполнения запроса),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени),
- 5400 - ERR_CALENDAR_MORE_DATA (размер массива недостаточен для получения описаний всех значений, поэтому получили только то, что вместилось),
- ошибки неудачного выполнения ArrayResize()

Примечание

Все функции для работы с Экономическим календарем используют время торгового сервера (TimeTradeServer). Это означает, что время в структуре MqlCalendarValue и входящие

© 2000-2025, MetaQuotes Ltd.

параметры времени в функциях CalendarValueHistoryByEvent/CalendarValueHistory задаются в таймзоне торгового сервера, а не в локальном времени пользователя.

Если в функцию был передан массив *events[]* фиксированной длины и в результате запроса не хватило места для сохранения всего результата, будет взведена ошибка ERR_CALENDAR_MORE_DATA (5400).

Если не задан параметр *datetime_to* (=0), то будут возвращены все значения событий от заданной даты *datetime_from*, имеющиеся в базе Календаря, в том числе и значения, которые запланированы на будущее.

Для фильтров *country_code* и *currency* значения NULL и "" равносильны - означают отсутствие фильтра.

Для *country_code* следует использовать поле *code* структуры MqlCalendarCountry, например "US", "RU" или "EU".

Для *currency* следует использовать поле *currency* структуры MqlCalendarCountry, например "USD", "RUB" или "EUR".

Фильтры применяются коньюкцией, т.е. через логическое 'И' выбираются значения только тех событий, для которых удовлетворяются одновременно оба условия - страна и валюта.

Структура MqlCalendarValue предоставляет методы для проверки и получения значений из полей *actual_value*, *forecast_value*, *prev_value* и *revised_prev_value*. Если значение поля не задано, то поле хранит значение LONG_MIN (-9223372036854775808).

При этом необходимо иметь в виду, что значения в этих полях хранятся увеличенными в миллион раз. Это означает, что при получении значений в MqlCalendarValue функциями CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent и CalendarValueLast, значения этих полей нужно проверять на равенство LONG_MIN, и если значение в поле задано, то для получения значения необходимо разделить значение поля на 1 000 000 (миллион). Другой способ получения значений - делать проверку и получать значения функциями самой структуры MqlCalendarValue.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- код страны для Европейского союза по стандарту ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- получим все значения событий по Европейскому союзу
   MqlCalendarValue values[];
//--- зададим границы диапазона, из которого берем события
   datetime date_from=D'01.01.2018';  // берем все события с 2018 года
   datetime date_to=0;                // 0 означает все известные события, даже те, чт
//--- запросим историю события по Европейскому союзу с 2018 года
   if(CalendarValueHistory(values,date_from,date_to,EU_code))
     {
      PrintFormat("Получены значения событий по country_code=%s: %d",
```

© 2000-2025, MetaQuotes Ltd.

```
                  EU_code,ArraySize(values));
      //--- уменьшим размер массива для вывода в Журнал
      ArrayResize(values,10);
//--- выведем значения событий в Журнал
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Ошибка! Не удалось получить события по стране country_code=%s",EU_c
      PrintFormat("Код ошибки: %d",GetLastError());
     }
//---
  }
/*
   Результат:
   Получены значения событий по 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       6060(
   [1] 54221  999500002 2018.01.04 09:00:00 2017.12.01 00:00:00          3       5660(
   [2] 54222  999500003 2018.01.04 09:00:00 2017.12.01 00:00:00          3       5810(
   [3] 45123  999030005 2018.01.05 10:00:00 2017.11.01 00:00:00          0         60(
   [4] 45124  999030006 2018.01.05 10:00:00 2017.11.01 00:00:00          0        280(
   [5] 45125  999030012 2018.01.05 10:00:00 2017.12.01 00:00:00          1         90(
   [6] 45126  999030013 2018.01.05 10:00:00 2017.12.01 00:00:00          1        140(
   [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        910(
   [9] 22231  999040004 2018.01.08 10:00:00 2017.12.01 00:00:00          0       1840(
*/
```

Смотри также

CalendarCountries, CalendarEventByCountry, CalendarValueHistoryByEvent, CalendarEventById, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueLastByEvent

Получает массив значений события по его ID с момента состояния базы Календаря с заданным change_id.

```
int  CalendarValueLastByEvent(
   ulong                event_id,      // идентификатор события
   ulong&               change_id,     // идентификатор значения события
   MqlCalendarValue&    values[]       // массив для получения описаний значений
   );
```

Параметры

event_id

[in] Идентификатор события.

change_id

[in][out] Идентификатор изменения.

values[]

[out] Массив типа MqlCalendarValue для получения значений события. Смотри пример обработки событий календаря.

Возвращаемое значение

Количество полученных значений события. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 4004 - ERR_NOT_ENOUGH_MEMORY (не достаточно памяти для выполнения запроса),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени),
- 5400 - ERR_CALENDAR_MORE_DATA (размер массива недостаточен для получения описаний всех значений, поэтому получили только то, что вместилось),
- ошибки неудачного выполнения ArrayResize()

Примечание

Все функции для работы с Экономическим календарем используют время торгового сервера (TimeTradeServer). Это означает, что время в структуре MqlCalendarValue и входящие параметры времени в функциях CalendarValueHistoryByEvent/CalendarValueHistory задаются в таймзоне торгового сервера, а не в локальном времени пользователя.

Если в функцию был передан массив *events[]* фиксированной длины и в результате запроса не хватило места для сохранения всего результата, будет взведена ошибка ERR_CALENDAR_MORE_DATA (5400).

Если в функцию передан *change_id* = 0, то функция всегда возвращает ноль, но при этом в *change_id* возвращается текущее состояние базы Календаря.

Функция возвращает массив значений для заданной новости и новое *change_id*, которые можно использовать для последующих вызовов данной функции, чтобы получить новые значения новости. Таким образом, вызывая эту функцию с последним известным *change_id*, можно получать обновления значений для заданной новости.

© 2000-2025, MetaQuotes Ltd.

Структура MqlCalendarValue предоставляет методы для проверки и получения значений из полей *actual_value*, *forecast_value*, *prev_value* и *revised_prev_value*. Если значение поля не задано, то поле хранит значение LONG_MIN (-9223372036854775808).

При этом необходимо иметь в виду, что значения в этих полях хранятся увеличенными в миллион раз. Это означает, что при получении значений в MqlCalendarValue функциями CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent и CalendarValueLast, значения этих полей нужно проверять на равенство LONG_MIN, и если значение в поле задано, то для получения значения необходимо разделить значение поля на 1 000 000 (миллион). Другой способ получения значений - делать проверку и получать значения функциями самой структуры MqlCalendarValue.

Пример эксперта, который слушает выход отчета по Nonfarm payrolls:

```
#property description "Пример использования функции CalendarValueLastByEvent"
#property description " для отлавливания выхода отчета по событию Nonfarm Payrolls."
#property description "Для этого необходимо получить текущий идентификатор изменения"
#property description " базы Календаря. И затем по этому идентификатору получать"
#property description " только новые события для через опрос в таймере"
//+------------------------------------------------------------------+
//| 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()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- идентификатор изменения базы Календаря
   static ulong calendar_change_id=0;
//--- признак первого запуска
   static bool first=true;
//--- идентификатор события
   static ulong event_id=0;
//--- имя события
   static string event_name=NULL;
//--- массив значений событий
   MqlCalendarValue values[];
//--- проведем инициализацию - получим текущий calendar_change_id
   if(first)
     {
      MqlCalendarEvent events[];
      //--- код страны для США по стандарту ISO 3166-1 Alpha-2
      string USA_code="US";
      //--- получим события для США
      int events_count=CalendarEventByCountry(USA_code,events);
      //--- позиция нужного нам события в массиве events
      int event_pos=-1;
      //--- выведем события для США в Журнал
      if(events_count>0)
        {
         PrintFormat("%s: События для США: %d",__FUNCTION__,events_count);
         for(int i=0;i<events_count;i++)
           {
            string event_name_low=events[i].name;
            //--- приведем к нижнему регистру имя события
            if(!StringToLower(event_name_low))
              {
               PrintFormat("StringToLower() вернула ошибку %d",GetLastError());
               //--- досрочно выходим из функции
               return;
              }
            //--- поищем событие "Nonfarm Payrolls"
            if(StringFind(event_name_low,"nonfarm payrolls")!=-1)
              {
               //--- событие найдено, запомним его id
               event_id=events[i].id;
               //--- запишем для события "Nonfarm Payrolls" его имя
               event_name=events[i].name;
               //--- запомним позицию событий в массиве events[]
               event_pos=i;
               //--- на самом деле в Календаре есть несколько событий, которые содержа
               PrintFormat("Событие \"Nonfarm Payrolls\" найдено: event_id=%d  event_n
               //--- посмотрите все события, закомментировав ниже оператор break, чтоб
               break;
              }
           }
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- сократим список, удалим события после события "Nonfarm Payrolls"
         ArrayRemove(events,event_pos+1);
         //--- для удобства изучения оставим 9 событий перед "Nonfarm Payrolls"
         ArrayRemove(events,0,event_pos-9);
         ArrayPrint(events);
        }
      else
        {
         PrintFormat("%s: CalendarEventByCountry(%s) вернуло 0 событий, код ошибки=%d"
                     USA_code,__FUNCTION__,GetLastError());
         //--- неудачное завершение работы, попробуем заново при следующем вызове тайм
         return;
        }

      //--- получим для указанного события идентификатор изменения базы Календаря
      if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
        {
         //--- этот блок кода не может выполниться при первом запуске, но мы его все р
         PrintFormat("%s: Получен текущий идентификатор базы Календаря: change_id=%d",
                     __FUNCTION__,calendar_change_id);
         //--- выставим флаг и выйдем до следущего события таймера
         first=false;
         return;
        }
      else
        {
         //--- данные не получены (для первого запуска это нормально), проверим налич
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: Получен текущий идентификатор базы Календаря: change_id=%
                        __FUNCTION__,calendar_change_id);
            //--- выставим флаг и выйдем до следущего события таймера
            first=false;
            //--- теперь у нас есть значение calendar_change_id
            return;
           }
         else
           {
            //--- а вот это действительно ошибка
            PrintFormat("%s: Не удалось получить значения для события event_id=%d",__F
            PrintFormat("Код ошибки: %d",error_code);
            //--- неудачное завершение работы, попробуем заново при следующем вызове т
            return;
           }
        }
     }

//--- у нас есть последнее известное значение идентификатора изменения Календаря (char
```

© 2000-2025, MetaQuotes Ltd.

```
   ulong old_change_id=calendar_change_id;
//--- проверим - не появилось ли новое значение события "Nonfarm Payrolls"
   if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
     {
      PrintFormat("%s: Получены новые события для \"%s\": %d",
                  __FUNCTION__,event_name,ArraySize(values));
      //--- выведем в Журнал информацию из массива values
      ArrayPrint(values);
      //--- выведем в Журнал значения предыдущего и нового идентификатора Календаря
      PrintFormat("%s: Предыдущий change_id=%d, новый change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
/*
      пропишите здесь свой код, который будет обрабатывать публикацию данных по "Nonfa
      */
     }
//---
  }
/*
   Результат:
   OnTimer: События для США: 202
   Событие "Nonfarm Payrolls" найдено: event_id=840030016  event_name=Nonfarm Payrolls
            [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importan
   [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: Получен текущий идентификатор базы Календаря: change_id=33986560

*/
```

Смотри также

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueLast

Получает массив значений по всем событиям с фильтрацией по стране и/или валюте с момента состояния базы Календаря с заданным change_id.

```
int  CalendarValueLast(
   ulong&               change_id,              // идентификатор изменения
   MqlCalendarValue&    values[],               // массив для получения описаний значен
   const string         country_code=NULL,      // кодовое имя страны по ISO 3166-1 alp
   const string         currency=NULL           // кодовое наименование валюты страны
   );
```

Параметры

change_id

[in][out] Идентификатор изменения.

values[]

[out] Массив типа MqlCalendarValue для получения значений события. Смотри пример обработки событий календаря.

country_code=NULL

[in] Кодовое имя страны согласно ISO 3166-1 alpha-2

currency=NULL

[in] Кодовое наименование валюты страны.

Возвращаемое значение

Количество полученных значений события. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Возможные ошибки:

- 4001 - ERR_INTERNAL_ERROR (общая ошибка исполняющей системы),
- 4004 - ERR_NOT_ENOUGH_MEMORY (не достаточно памяти для выполнения запроса),
- 5401 - ERR_CALENDAR_TIMEOUT (превышен лимит запроса по времени),
- 5400 - ERR_CALENDAR_MORE_DATA (размер массива недостаточен для получения описаний всех значений, поэтому получили только то, что вместилось),
- ошибки неудачного выполнения ArrayResize()

Примечание

Все функции для работы с Экономическим календарем используют время торгового сервера (TimeTradeServer). Это означает, что время в структуре MqlCalendarValue и входящие параметры времени в функциях CalendarValueHistoryByEvent/CalendarValueHistory задаются в таймзоне торгового сервера, а не в локальном времени пользователя.

Если в функцию был передан массив *events[]* фиксированной длины и в результате запроса не хватило места для сохранения всего результата, будет взведена ошибка ERR_CALENDAR_MORE_DATA (5400).

Если в функцию передан *change_id* = 0, то функция всегда возвращает ноль, но при этом в *change_id* возвращается текущее состояние базы Календаря.

© 2000-2025, MetaQuotes Ltd.

Для фильтров *country_code* и *currency* значения NULL и "" равносильны - означают отсутствие фильтра.

Для *country_code* следует использовать поле *code* структуры MqlCalendarCountry, например "US", "RU" или "EU".

Для *currency* следует использовать поле *currency* структуры MqlCalendarCountry, например "USD", "RUB" или "EUR".

Фильтры применяются коньюкцией, т.е. через логическое 'И' выбираются значения только тех событий, для которых удовлетворяются одновременно оба условия - страна и валюта

Функция возвращает массив значений для заданной новости и новое *change_id*, которые можно использовать для последующих вызовов данной функции, чтобы получить новые значения новости. Таким образом, вызывая эту функцию с последним известным *change_id*, можно получать обновления значений для заданной новости.ы

Структура MqlCalendarValue предоставляет методы для проверки и получения значений из полей *actual_value*, *forecast_value*, *prev_value* и *revised_prev_value*. Если значение поля не задано, то поле хранит значение LONG_MIN (-9223372036854775808).

При этом необходимо иметь в виду, что значения в этих полях хранятся увеличенными в миллион раз. Это означает, что при получении значений в MqlCalendarValue функциями CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent и CalendarValueLast, значения этих полей нужно проверять на равенство LONG_MIN, и если значение в поле задано, то для получения значения необходимо разделить значение поля на 1 000 000 (миллион). Другой способ получения значений - делать проверку и получать значения функциями самой структуры MqlCalendarValue.

Пример эксперта, который слушает появление событий Экономического календаря:

```
#property description "Пример использования функции CalendarValueLast"
#property description " для создания слушателя событий Экономического календаря."
#property description "Для этого необходимо получить текущий идентификатор изменения"
#property description " базы Календаря. И затем по этому идентификатору получать"
#property description " только новые события для через опрос в таймере"
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create timer
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy timer
   EventKillTimer();
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- идентификатор изменения базы Календаря
   static ulong calendar_change_id=0;
//--- признак первого запуска
   static bool first=true;
//--- массив значений событий
   MqlCalendarValue values[];
//--- проведем инициализацию - получим текущий calendar_change_id
   if(first)
     {
      //--- получим идентификатор изменения базы Календаря
      if(CalendarValueLast(calendar_change_id,values)>0)
        {
         //--- этот блок кода не может выполниться при первом запуске, но мы его все р
         PrintFormat("%s: Получен текущий идентификатор базы Календаря: change_id=%d",
                     __FUNCTION__,calendar_change_id);
         //--- выставим флаг и выйдем до следущего события таймера
         first=false;
         return;
        }
      else
        {
         //--- данные не получены (для первого запуска это нормально), проверим наличи
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: Получен текущий идентификатор базы Календаря: change_id=%
                        __FUNCTION__,calendar_change_id);
            //--- выставим флаг и выйдем до следущего события таймера
            first=false;
            //--- теперь у нас есть значение calendar_change_id
            return;
           }
         else
           {
            //--- а вот это действительно ошибка
```

© 2000-2025, MetaQuotes Ltd.

```
            PrintFormat("%s: Не удалось получить события в CalendarValueLast. Код ошиб
                        __FUNCTION__,error_code);
            //--- неудачное завершение работы, попробуем провести инициализацию заново
            return;
           }
        }
     }

//--- у нас есть последнее известное значение идентификатора изменения Календаря (char
   ulong old_change_id=calendar_change_id;
//--- проверим - не появились ли новые события Календаря
   if(CalendarValueLast(calendar_change_id,values)>0)
     {
      PrintFormat("%s: Получены новые события Календаря: %d",
                  __FUNCTION__,ArraySize(values));
      //--- выведем в Журнал информацию из массива values
      ArrayPrint(values);
      //--- выведем в Журнал значения предыдущего и нового идентификатора Календаря
      PrintFormat("%s: Предыдущий change_id=%d, новый change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
      //--- выведем в Журнал новые события
      ArrayPrint(values);
      /*
      пропишите здесь свой код, который будет обрабатывать появление событий
      */
     }
//---
  }
/*
   Пример работы слушателя:
   OnTimer: Получен текущий идентификатор базы Календаря: change_id=33281792
   OnTimer: Получены новые события для Календаря: 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       -507
   OnTimer: Предыдущий change_id=33281792, новый 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       -507
   OnTimer: Получены новые события для Календаря: 1
        [id] [event_id]              [time]            [period] [revision]       [actu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -922337203
   OnTimer: Предыдущий change_id=33282048, новый 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 -922337203

*/
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

Доступ к таймсериям и данным индикаторов

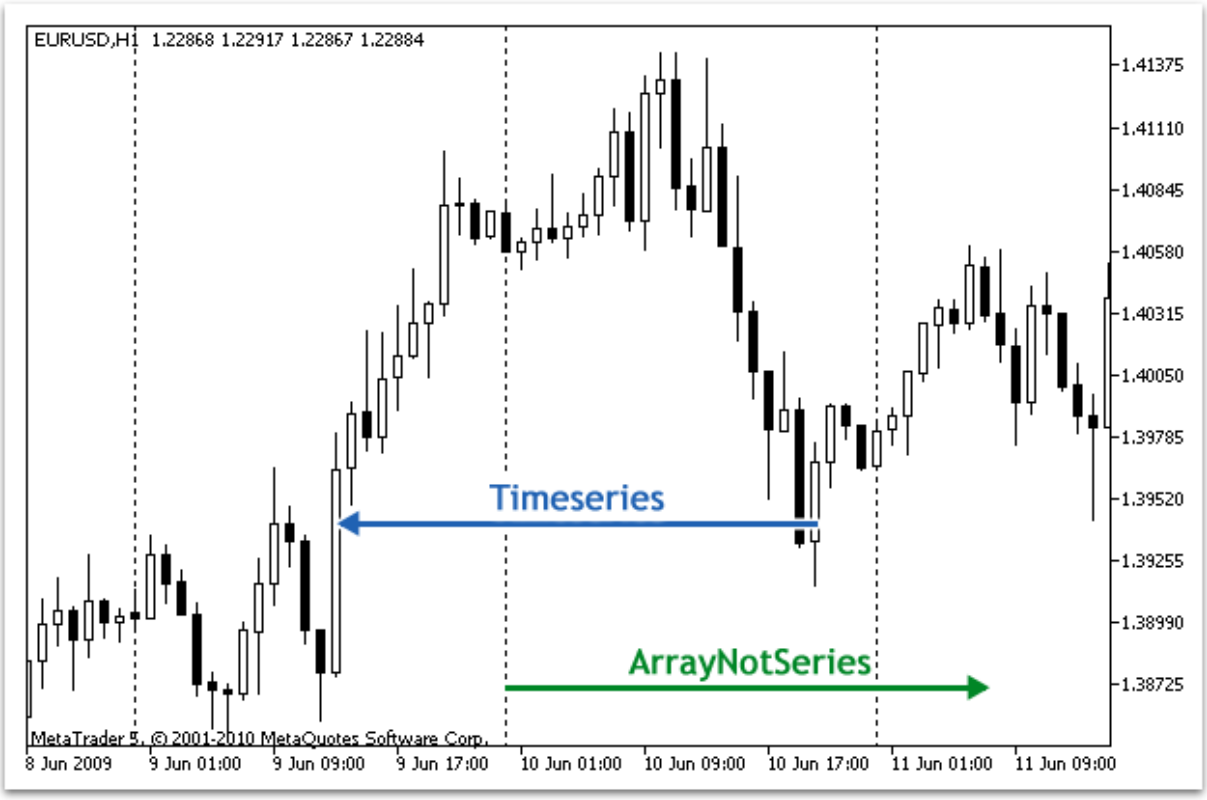
Функции для работы с таймсериями и индикаторами. Таймсерия отличается от обычного массива тем, что индексация элементов таймсерии производится от конца массива к началу (от самых свежих данных к самым старым). Для копирования значений таймсерий и индикаторов рекомендуется использовать только динамические массивы, так как функции копирования самостоятельно распределяют необходимый размер массивов-приемников значений.

Из этого правила есть **важное исключение**: если копирование таймсерий и значений индикаторов необходимо делать часто, например, при каждом вызове OnTick() в экспертах или при каждом вызове OnCalculate() в индикаторах, то в этом случае лучше использовать статически распределенные массивы, так как **операции распределения памяти** под динамические массивы **требуют дополнительного времени** и это скажется при тестировании и оптимизации экспертов.

При использовании функций доступа к таймсериям и значениям индикаторов необходимо учитывать направление индексации, это подробно описано в разделе Направление индексации в массивах и таймсериях.

Доступ к данным индикаторов и таймсерий осуществляется независимо от факта готовности запрашиваемых данных (так называемый асинхронный доступ). Это критически важно для расчета пользовательских индикаторов, поэтому при отсутствии запрашиваемых данных функции типа *Copy...()* сразу же возвращают ошибку. Однако при доступе из экспертов и скриптов производится несколько попыток получения данных с небольшой паузой, призванной обеспечить время, необходимое для загрузки недостающих таймсерий либо для расчета значений индикаторов.

В разделе Организация доступа к данным дается подробное описание тонкостей получения, хранения и запроса ценовых данных в клиентском терминале MetaTrader 5.



© 2000-2025, MetaQuotes Ltd.

Исторически сложилось так, что доступ к данным в ценовом массиве производился с конца данных. Физически новые данные всегда дописываются в конец массива, но индекс этого массива всегда равен нулю. Индекс 0 в массиве-таймсерии означает данные текущего бара, то есть бара, который соответствует незавершенному промежутку времени на данном таймфрейме.

Таймфрейм – период времени, в течение которого формируется один ценовой бар; всего предопределен 21 стандартный таймфрейм.

Функция	Действие
SeriesInfoInteger	Возвращает информацию о состоянии исторических данных
Bars	Возвращает количество баров в истории по соответствующим символу и периоду
BarsCalculated	Возвращает количество рассчитанных данных в индикаторном буфере или -1 в случае ошибки (данные еще не рассчитаны)
IndicatorCreate	Возвращает хэндл указанного технического индикатора, созданного на основе массива параметров типа MqlParam
IndicatorParameters	Возвращает по указанному хэндлу количество входных параметров индикатора, а также сами значения и тип параметров
IndicatorRelease	Удаляет хэндл индикатора и освобождает расчетную часть индикатора, если ею больше никто не пользуется
CopyBuffer	Получает в массив данные указанного буфера от указанного индикатора
CopyRates	Получает в массив исторические данные структуры Rates для указанных символа и периода
CopySeries	Получает в указанный набор массивов синхронизированные таймсерии из структуры Rates для указанного символа-периода в указанном количестве
CopyTime	Получает в массив исторические данные по времени открытия баров по соответствующим символу и периоду
CopyOpen	Получает в массив исторические данные по цене открытия баров по соответствующим символу и периоду
CopyHigh	Получает в массив исторические данные по максимальной цене баров по соответствующим символу и периоду
CopyLow	Получает в массив исторические данные по минимальной цене баров по соответствующим символу и периоду
CopyClose	Получает в массив исторические данные по цене закрытия баров по соответствующим символу и периоду
CopyTickVolume	Получает в массив исторические данные по тиковым объемам для соответствующих символа и периода
CopyRealVolume	Получает в массив исторические данные по торговым объемам для соответствующих символа и периода

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
CopySpread	Получает в массив исторические данные по спредам для соответствующих символа и периода
CopyTicks	Получает в массив тики в формате MqlTick
CopyTicksRange	Получает в массив тики в указанном диапазоне дат
iBars	Возвращает количество баров в истории по соответствующему символу и периоду
iBarShift	Возвращает индекс бара, в который попадает указанное время
iClose	Возвращает значение цены закрытия бара (указанного параметром shift) соответствующего графика
iHigh	Возвращает значение максимальной цены бара (указанного параметром shift) соответствующего графика
iHighest	Возвращает индекс наибольшего найденного значения (смещение относительно текущего бара) соответствующего графика
iLow	Возвращает значение минимальной цены бара (указанного параметром shift) соответствующего графика
iLowest	Возвращает индекс наименьшего найденного значения (смещение относительно текущего бара) соответствующего графика
iOpen	Возвращает значение цены открытия бара (указанного параметром shift) соответствующего графика
iTime	Возвращает значение времени открытия бара (указанного параметром shift) соответствующего графика
iTickVolume	Возвращает значение тикового объема бара (указанного параметром shift) соответствующего графика
iRealVolume	Возвращает значение реального объема бара (указанного параметром shift) соответствующего графика
iVolume	Возвращает значение тикового объема бара (указанного параметром shift) соответствующего графика
iSpread	Возвращает значение спреда бара (указанного параметром shift) соответствующего графика

Несмотря на то, что функцией ArraySetAsSeries() можно задавать массивам способ доступа к элементам как для таймсерии, нужно помнить, что физически элементы массива всегда хранятся в одном и том же порядке, меняется только направление индексации. Для демонстрации этого факта можно выполнить пример:

```
   datetime TimeAsSeries[];
//--- установим доступ к массиву как к таймсерии
   ArraySetAsSeries(TimeAsSeries,true);
   ResetLastError();
   int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(copied<=0)
     {
      Print("Не удалось скопировать время открытия для последних 10 баров");
      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("Не удалось скопировать время открытия для последних 10 баров");
      return;
     }
   size=ArraySize(ArrayNotSeries);
   for(int i=size-1;i>=0;i--)
     {
      Print("ArrayNotSeries["+i+"] = ",ArrayNotSeries[i]);
     }
```

В результате будет произведен вывод подобный этому:

```
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
ArrayNotSeries[3] = 2009.06.11 08:00:00
```

© 2000-2025, MetaQuotes Ltd.

```
ArrayNotSeries[2] = 2009.06.11 07:00:00
ArrayNotSeries[1] = 2009.06.11 06:00:00
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

Как видно из результатов вывода, для массива TimeAsSeries с ростом индекса уменьшается значение времени, находящегося под этим индексом, то есть мы продвигаемся от настоящего к прошлому. Для обычного массива ArrayNotSeries все наоборот - с ростом индекса мы двигаемся из прошлого к настоящему.

Смотри также

ArrayIsDynamic, ArrayGetAsSeries, ArraySetAsSeries, ArrayIsSeries

© 2000-2025, MetaQuotes Ltd.

Направление индексации в массивах, буферах и таймсериях

Все массивы и индикаторные буферы по умолчанию имеют направление индексации слева направо. Индекс первого элемента всегда равен нулю. Таким образом, самый первый элемент массива или индикаторного буфера с индексом 0 по умолчанию находится на крайней левой позиции, последний элемент находится на крайней правой позиции.

Индикаторный буфер представляет из себя динамический массив типа double, размером которого управляет клиентский терминал с тем, чтобы он всегда соответствовал количеству баров, на которых индикатор рассчитывается. Обычный динамический массив типа double назначается в качестве индикаторного буфера с помощью функции SetIndexBuffer(). Для индикаторных буферов не требуется задавать размер с помощью функции ArrayResize(), исполняющая система терминала сама позаботится об этом.

Таймсерии представляют из себя массивы с обратной индексацией, то есть самый первый элемент таймсерии находится на крайней правой позиции, а последний элемент таймсерии находится на крайней левой позиции. Так как таймсерии предназначены для хранения исторических ценовых данных по финансовым инструментам и обязательно содержат информацию о времени, то можно сказать, что самые свежие данные в таймсерии находятся в правой крайней позиции, а самые старые в крайней левой позиции.

Поэтому, элемент с индексом ноль в таймсерии содержит информацию о самой последней котировке по инструменту. Если таймсерия представляет данные по дневному таймфрейму, то на нулевой позиции содержатся данные текущего незавершенного дня, а на позиции с индексом один хранятся данные вчерашнего дня.

Изменение направления индексации

Функция ArraySetAsSeries() позволяет изменять способ доступа к элементам динамического массива, но при этом физически порядок хранения данных в памяти компьютера не изменяется. Эта функция просто изменяет способ адресации к элементам массива, поэтому при копировании одного массива в другой с помощью функции ArrayCopy() содержимое массива-приемника не будет зависеть от направления индексации в массиве-источнике.

Нельзя изменять направление индексации для статически распределенных массивов. Даже если массив был передан в качестве параметра в функцию, то и внутри этой функции попытки изменения направления индексации ни к чему не приведут.

Для индикаторных буферов, как и для обычных массивов, также разрешается устанавливать направление индексации задом наперед как в таймсерии, то есть, обращение к нулевой позиции в индикаторном буфере в этом случае будет означать обращение к самому последнему значению в соответствующем индикаторном буфере и это будет соответствовать значению индикатора на самом последнем баре. При этом физически размещение данных в индикаторном буфере останется неизменным, как уже упоминалось.

Получение ценовых данных в индикаторах

В каждом пользовательском индикаторе обязательно должна присутствовать функция OnCalculate(), которой передаются ценовые данные, необходимые для расчета значений в

© 2000-2025, MetaQuotes Ltd.

индикаторных буферах. Направление индексации в этих переданных массивах можно выяснить с помощью функции ArrayGetAsSeries().

Переданные в функцию массивы отражают ценовые данные, т.е. эти массивы имеют признак таймсерии и функция ArrayIsSeries() вернет true при проверке этих массивов. Но тем не менее, направление индексации необходимо в любом случае проверять только функцией ArrayGetAsSeries().

Чтобы не зависеть от умолчаний, необходимо безусловно вызывать функцию ArraySetAsSeries() для тех массивов, с которыми предполагается работать, и установить требуемое направление индексации.

Получение ценовых данных и значений индикаторов

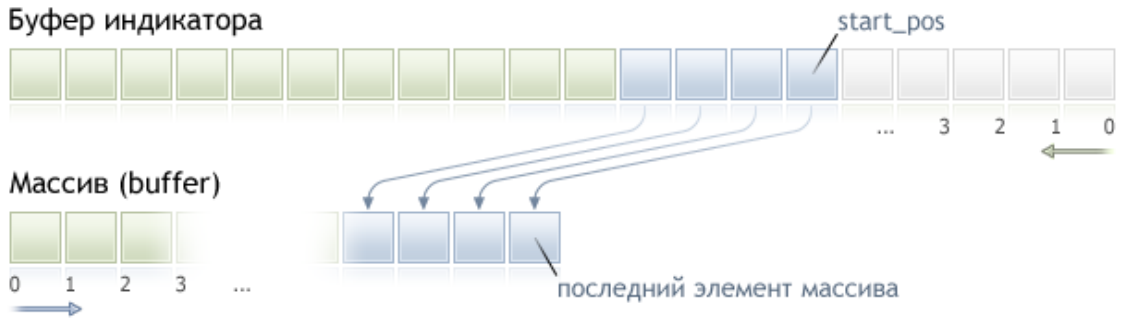
В экспертах, индикаторах и скриптах все массивы по умолчанию имеют направление индексации слева направо. При необходимости в любой mql5-программе можно запросить значения таймсерий по любому символу и таймфрейму, а также значения индикаторов, рассчитанных на любом символе и таймфрейме.

Для получения таких данных служат функции Copy...():

- CopyBuffer - копирование значений индикаторного буфера в массив типа double;
- CopyRates - копирование ценовой истории в массив структур MqlRates;
- CopyTime - копирование значений Time в массив типа datetime;
- CopyOpen - копирование значений Open в массив типа double;
- CopyHigh - копирование значений High в массив типа double;
- CopyLow - копирование значений Low в массив типа double;
- CopyClose - копирование значений Close в массив типа double;
- CopyTickVolume - копирование тиковых объемов в массив типа long;
- CopyRealVolume - копирование биржевых объемов в массив типа long;
- CopySpread - копирование истории спредов в массив типа int;

Все эти функции работают одинаково, и поэтому достаточно рассмотреть механизм получения данных на примере CopyBuffer(). Подразумевается, что все запрашиваемые данные имеют направление индексации, как в таймсерии, при этом подразумевается, что в позиции с индексом 0 (ноль) хранятся данные текущего незавершенного бара. Чтобы получить доступ к этим данным, нам необходимо скопировать нужный нам объем данных в массив-приемник, например, в массив *buffer*.

© 2000-2025, MetaQuotes Ltd.



При копировании необходимо указать стартовую позицию в массиве-источнике, начиная с которой будут копироваться данные в массив-приемник. В случае успеха в массив-получатель будет скопировано указанное количество элементов из массива-источник, в данном случае из индикаторного буфера. При этом независимо от того, какое направление индексации установлено в массиве получателе, копирование всегда производится так, как указано на рисунке.

Если ценовые данные предполагается обрабатывать в цикле с большим количество итераций, то рекомендуется проверять факт принудительного завершения программы с помощью функции IsStopped():

```
int copied=CopyBuffer(ma_handle,// хэндл индикатора
                      0,        // индекс индикаторного буфера
                      0,        // стартовая позиция для копирования
                      number,   // количество значений для копирования
                      Buffer    // массив-получатель значений
                      );
if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
  {
   //--- получаем значение для индекса k
   double value=Buffer[k];
   // ...
   // работа со значением value
   k++;
  }
```

Пример:

```
input int per=10; // период экспоненты
int ma_handle;    // хэндл индикатора
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
   double ema[10];
   int copied=CopyBuffer(ma_handle,// хэндл индикатора
                         0,        // индекс индикаторного буфера
                         0,        // стартовая позиция для копирования
                         10,       // количество значений для копирования
                         ema       // массив-получатель значений
                         );
   if(copied<0) return;
// .... дальнейший код
  }
```

Смотри также

[Организация доступа к данным]

© 2000-2025, MetaQuotes Ltd.

Организация доступа к данным

В этом разделе рассматриваются вопросы, связанные с получением, хранением и запросами ценовых данных (таймсерий).

Получение данных от торгового сервера

Прежде чем ценовые данные будут доступны в терминале MetaTrader 5, их необходимо получить и обработать. Для получения данных требуется подключение к торговому серверу MetaTrader 5. Данные поступают с сервера по запросу терминала в виде экономно упакованных блоков минутных баров.

Механизм обращения к серверу за данными не зависит от того, каким образом был инициирован запрос — пользователем при навигации по графику или программным способом на языке MQL5.

Хранение промежуточных данных

Полученные с сервера данные автоматически распаковываются и сохраняются в специальном промежуточном формате HCC. Данные по каждому символу пишутся в отдельную папку *каталог_терминала*\bases*имя_сервера*\history*имя_символа*. Например, данные по символу EURUSD с торгового сервера MetaQuotes-Demo будут находиться в папке *каталаг_терминала*\bases\MetaQuotes-Demo\history\EURUSD\.

Данные записываются в файлы с расширением .hcc, каждый файл хранит данные минутных баров за год. Например, файл 2009.hcc в папке EURUSD содержит минутные бары по символу EURUSD за 2009 год. Эти файлы используются для подготовки ценовых данных по всем таймфреймам и не предназначены для прямого доступа.

Получение данных нужного таймфрейма из промежуточных данных

Служебные файлы в формате HCC исполняют роль источника данных для построения ценовых данных по запрошенным таймфреймам в формате HC. Данные в формате HC являются таймсериями, максимально подготовленными для быстрого доступа. Они создаются только по запросу графика или mql5-программы в объеме, не превышающем значения параметра "Max bars in charts", и сохраняются для дальнейшего использования в файлах с расширением hc.

Для экономии ресурсов данные по таймфрейму загружаются и хранятся в оперативной памяти только по необходимости, при длительном отсутствии обращений к данным происходит выгрузка их из оперативной памяти с сохранением в файл. Для каждого таймфрейма данные подготавливаются независимо от наличия уже готовых данных для других таймфреймов. Правила формирования и доступности данных одинаковы для всех таймфреймов. Т.е. не смотря на то, что единицей хранения данных в формате HCC является минутный бар, наличие данных в формате HCC не означает наличие и доступность в том же объеме данных таймфрейма M1 в формате HC.

Получение новых данных с сервера вызывает автоматическое обновление используемых ценовых данных в формате HC по всем таймфреймам и перерасчет всех индикаторов, которые явно используют их в качестве входных данных для расчета.

Параметр "Max bars in chart"

© 2000-2025, MetaQuotes Ltd.

Параметр "Max bars in charts" ограничивает доступное для графиков, индикаторов и mql5-программ количество баров в формате HC. Это ограничение действует для данных всех таймфреймов, и предназначено в первую очередь для экономии ресурсов.

Устанавливая большие значения данного параметра, следует помнить, что при наличии достаточно глубокой истории ценовых данных для младших таймфреймов расход памяти на хранение таймсерий и буферов индикаторов может составить сотни мегабайт и достигнуть ограничения оперативной памяти для программы клиентского терминала (2Гб для 32-битных приложений MS Windows).

Изменение параметра "Max bars in charts" вступает в силу после перезапуска клиентского терминала. Само по себе изменение данного параметра не вызывает ни автоматического обращения к серверу за дополнительными данными, ни формирования дополнительных баров таймсерий. Запрос дополнительных ценовых данных у сервера и обновление таймсерий с учетом нового ограничения произойдет либо в случае прокрутки графика в область недостающих данных, либо в случае запроса недостающих данных из mql5-программы.

Объем запрашиваемых у сервера данных соответствует требуемому количеству баров данного таймфрейма с учетом значения параметра "Max bars in charts". Ограничение, задаваемое параметром, не является жестким, и в некоторых случаях количество доступных баров по таймфрейму может быть незначительно больше текущего значения параметра.

Доступность данных

Наличие данных в формате HCC или даже в готовом для использования формате HC не всегда означает безусловную доступность этих данных для отображения на графике или для использования в mql5-программах.

При доступе к ценовым данным или к значениям индикаторов из mql5-программ следует помнить, что не гарантируется их доступность в определенный момент времени, либо с определенного момента времени. Это связано с тем, что в целях экономии ресурсов в MetaTrader 5 не хранится полная копия требуемых данных для mql5-программы, а дается прямой доступ к базе данных терминала.

Ценовая история по всем таймфреймам строится из общих данных формата HCC и любое обновление данных с сервера приводит к обновлению данных по всем таймфреймам и пересчету индикаторов. Вследствие этого, в доступе к данным может быть отказано даже в том случае, если эти данные были доступны мгновение назад.

Синхронизация данных терминала и данных сервера

Поскольку mql5-программа может обратиться к данным по любому символу и таймфрейму, то есть вероятность, что данные требуемой таймсерии еще не сформированы в терминале или требуемые ценовые данные не синхронизированы с торговым сервером. В этом случае время ожидания готовности данных сложно прогнозировать.

Алгоритмы с использованием циклов ожидания готовности данных являются не лучшим решением. Единственное исключение в данном случае — скрипты, так как у них нет другого выбора алгоритма в виду отсутствия обработки событий. Для пользовательских индикаторов подобные алгоритмы, как и любые другие циклы ожидания, категорически не рекомендуются, так как приводят к остановке расчета всех индикаторов и другой обработки ценовых данных по данному символу.

© 2000-2025, MetaQuotes Ltd.

Для экспертов и пользовательских индикаторов лучше использовать событийную модель обработки. Если при обработке события OnTick() или OnCalculate() не удалось получить все необходимые данные требуемой таймсерии, то следует выйти из обработчика события, рассчитывая на появление доступа к данным при следующем вызове обработчика.

Пример скрипта для закачки истории

Рассмотрим пример скрипта, который выполняет запрос на получение истории с торгового сервера по указанному инструменту. Скрипт предназначен для запуска на графике требуемого инструмента, таймфрейм значения не имеет, так как, как уже было сказано выше, ценовые данные приходят с торгового сервера в виде упакованных минутных данных, из которых в последствие и строится любая предопределенная таймсерия.

Оформим все действия по получению данных в виде отдельной функции CheckLoadHistory(symbol, timeframe, start_date):

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
  }
```

Функция CheckLoadHistory() задумана как универсальная, которую могут вызвать из любой программы (эксперт, скрипт или индикатор), и поэтому требует три входных параметра: имя символа, период и начальную дату, с которой нам требуется ценовая история.

Вставим в код функции все необходимые проверки, прежде чем мы потребуем недостающую историю. Первым делом необходимо убедиться, что имя символа и значение периода являются корректными:

```
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
```

Далее — убедимся, что указанный символ доступен в окне MarketWatch, то есть, история по данному символу будет доступна при запросе к торговому серверу. Если его там нет - добавим символ в окно самостоятельно с помощью функции SymbolSelect().

```
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
```

Теперь необходимо получить начальную дату по уже имеющейся истории для указанной пары символ/период. Возможно, что значение входного параметра startdate, переданного функции CheckLoadHistory() попадает в интервал уже доступной истории, и тогда никакого запроса к торговому серверу не потребуется. Для получения самой первой даты по символу-периоду на данный момент предназначена функция SeriesInfoInteger() с модификатором SERIES_FIRSTDATE.

```
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

Следующая важная проверка — проверка типа программы, из которой вызывается функция. Напомним, что отправка запроса на обновление таймсерии с тем же периодом, что и у индикатора, вызывающего обновление, крайне нежелательна. Нежелательность запроса данных

© 2000-2025, MetaQuotes Ltd.

по тому же символу-периоду, что и у индикатора обусловлена тем, что обновление исторических данных производится в том же потоке, в котором работает индикатор. Поэтому велика вероятность клинча. Для проверки используем функцию MQL5InfoInteger() с модификатором MQL5_PROGRAM_TYPE.

```
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sy
      return(-4);
```

Если мы успешно прошли все проверки, то сделаем последнюю попытку обойтись без обращения к торговому серверу. Сначала узнаем начальную дату, для которой доступны минутные данные в формате HCC. Запросим это значение функцией SeriesInfoInteger() с модификатором SERIES_TERMINAL_FIRSTDATE и опять сравним со значением параметра start_date.

```
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- there is loaded data to build timeseries
      if(first_date>0)
        {
         //--- force timeseries build
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- check date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
```

Если после всех проверок поток выполнения по-прежнему находится в теле функции CheckLoadHistory(), то значит, есть необходимость запросить недостающие ценовые данные с торгового сервера. Для начала мы узнаем значение "Max bars in chart" с помощью функции TerminalInfoInteger():

```
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

Оно нам понадобится для того, чтобы не запрашивать лишние данные. Затем выясним самую первую дату в истории по символу на торговом сервере (независимо от периода) посредством уже знакомой функции SeriesInfoInteger() с модификатором SERIES_SERVER_FIRSTDATE.

```
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
```

Так как запрос является асинхронной операцией, то функция вызывается в цикле с небольшой задержкой в 5 миллисекунд до тех пор, пока переменная first_server_date не получит значение либо выполнение цикла не будет прервано пользователем (IsStopped() в этом случае вернет значение true). Укажем корректное значение начальной даты, начиная с которой мы запрашиваем у торгового сервера ценовые данные.

```
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Warning: first server date ",first_server_date," for ",symbol,
            " does not match to first series date ",first_date);
```

© 2000-2025, MetaQuotes Ltd.

Если вдруг начальная дата *first_server_date* на сервере окажется меньше, чем начальная дата *first_date* по символу в формате HCC , то в журнал будет выведено соответствующее сообщение.

Теперь мы готовы сделать запрос к торговому серверу за недостающими ценовыми данными. Запрос сделаем в виде цикла и начнем заполнять его тело:

```
   while(!IsStopped())
     {
      //1. дождаться синхронизации между перестроенной таймсерией и промежуточной исто
      //2. получить текущее количество баров bars на данной таймсерии
      //    если bars больше, чем Max_bars_in_chart, то можем выходить, работа окончен
      //3. Получим начальную дату first_date в перестроенной таймсерии и сравним со зн
      //    если first_date меньше чем start_date, то можем выходить, работа окончена
      //4. Запросим новую порцию истории в 100 баров у торгового сервера от последнего
     }
```

Первые три пункта реализуются уже знакомыми приемами.

```
   while(!IsStopped())
     {
      //--- 1.дождаться окончания процесса перестройки таймсерии
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- 2.запросим сколько баров мы теперь имеем
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         //--- баров больше, чем можно отобразить на графике, выходим
         if(bars>=max_bars) return(-2);
         //--- 3. узнаем текущую начальную дату в таймсерии
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            // начальная дата более ранняя, чем запрашивалось, задача выполнена
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //4. Запросим новую порцию истории в 100 баров у торгового сервера от последнего
     }
```

Остался последний четвертый пункт — непосредственный запрос истории. Мы не можем прямо обратиться к серверу, но любая Copy-функция при нехватке истории в формате HCC терминал автоматически инициирует посылку такого запроса от терминала к торговому серверу. Так как время самой первой начальной даты в переменной *first_date* является самым простым и естественным критерием для оценки степени выполнения запроса, то самым простым будет использовать функцию CopyTime().

При вызове функций, осуществляющих копирование любых данных из таймсерий, необходимо иметь в виду то, что параметр *start* (номер бара, с которого начинать копирование ценовых данных) всегда должен быть в пределах доступной истории терминала. Если у нас имеется только 100 баров, то не имеет смысла пытаться скопировать 300 баров, начиная с бара с индексом 500. Такой запрос будет воспринят как ошибочный и не будет обработан, т.е. никакая история с торгового сервера подгружена не будет.

© 2000-2025, MetaQuotes Ltd.

Именно поэтому мы будем копировать по 100 баров, начиная с бара с индексом bars. Это обеспечит плавную подкачку истории с торгового сервера, при этом реально будет подгружено чуть более запрошенных 100 баров, сервер отдает историю с запасом.

```
   int copied=CopyTime(symbol,period,bars,100,times);
```

После операции копирования необходимо проанализировать количество скопированных элементов, если попытка оказалась неудачной, то значение переменной *copied* будет равно нулю и значение счетчика fail_cnt будет увеличено на 1. После 100 неудачных попыток работа функции будет прервана.

```
int fail_cnt=0;
...
   int copied=CopyTime(symbol,period,bars,100,times);
   if(copied>0)
     {
      //--- проверим данные
      if(times[0]<=start_date)  return(0);  // скопированное значение меньше, готово
      if(bars+copied>=max_bars) return(-2); // баров стало больше, чем помещается на г
      fail_cnt=0;
     }
   else
     {
      //--- не более 100 неудачных попыток подряд
      fail_cnt++;
      if(fail_cnt>=100) return(-5);
      Sleep(10);
     }
```

Таким образом, в функции не только организована правильная обработка текущей ситуации в каждый момент выполнения, но и еще возвращается код завершения, который мы можем обработать после вызова функции CheckLoadHistory() для получения дополнительную информацию. Например, таким образом:

```
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Неизвестный символ",InpLoadedSymbol);                        br
      case -2 : Print("Запрошенных баров больше, чем можно отобразить на графике"); br
      case -3 : Print("Выполнение было прервано пользователем");                    br
      case -4 : Print("Индикатор не должен загружать собственные данные");         br
      case -5 : Print("Загрузка окончилась неудачей");                             br
      case  0 : Print("Все данные загружены");                                     br
      case  1 : Print("Уже имеющихся данных в таймсерии достаточно");              br
      case  2 : Print("Таймсерия построена из имеющихся данных терминала");        br
      default : Print("Результат выполнения не определен");
     }
```

Полный код функции приведен в примере скрипта, демонстрирующего правильный способ организации доступа к любым данным с обработкой результат запроса.

© 2000-2025, MetaQuotes Ltd.

Код:

```
//+------------------------------------------------------------------+
//|                                                TestLoadHistory.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- input parameters
input string          InpLoadedSymbol="NZDUSD";   // Symbol to be load
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1;  // Period to be load
input datetime        InpStartDate=D'2006.01.01'; // Start date
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
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("First date ",first_date," - ",bars," bars");
//---
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
   datetime first_date=0;
   datetime times[100];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- check symbol & period
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
//--- check if symbol is selected in the MarketWatch
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
//--- check if data is present
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
//--- don't ask for load of its own data if it is an indicator
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sy
      return(-4);
//--- second attempt
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- there is loaded data to build timeseries
      if(first_date>0)
        {
         //--- force timeseries build
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- check date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
//--- max bars in chart from terminal options
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- load symbol history info
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
//--- fix start date for loading
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Warning: first server date ",first_server_date," for ",symbol,
            " does not match to first series date ",first_date);
//--- load data step by step
   int fail_cnt=0;
   while(!IsStopped())
     {
      //--- wait for timeseries build
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- ask for built bars
      int bars=Bars(symbol,period);
      if(bars>0)
```

© 2000-2025, MetaQuotes Ltd.

```
        {
         if(bars>=max_bars) return(-2);
         //--- ask for first date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //--- copying of next part forces data loading
      int copied=CopyTime(symbol,period,bars,100,times);
      if(copied>0)
        {
         //--- check for data
         if(times[0]<=start_date)  return(0);
         if(bars+copied>=max_bars) return(-2);
         fail_cnt=0;
        }
      else
        {
         //--- no more than 100 failed attempts
         fail_cnt++;
         if(fail_cnt>=100) return(-5);
         Sleep(10);
        }
     }
//--- stopped
   return(-3);
  }
//+------------------------------------------------------------------+
//| Возвращает строкое значение периода                              |
//+------------------------------------------------------------------+
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");
```

© 2000-2025, MetaQuotes Ltd.

```
      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");
      case PERIOD_MN1: return("Monthly");
     }
//---
   return("unknown period");
  }
```

© 2000-2025, MetaQuotes Ltd.

SeriesInfoInteger

Возвращает информацию о состоянии исторических данных. Существует 2 варианта функции.

Непосредственно возвращает значение свойства.

```
long  SeriesInfoInteger(
   string                      symbol_name,      // имя символа
   ENUM_TIMEFRAMES             timeframe,        // период
   ENUM_SERIES_INFO_INTEGER    prop_id,          // идентификатор свойства
   );
```

Возвращает true или false в зависимости от успешности выполнения функции.

```
bool  SeriesInfoInteger(
   string                      symbol_name,      // имя символа
   ENUM_TIMEFRAMES             timeframe,        // период
   ENUM_SERIES_INFO_INTEGER    prop_id,          // идентификатор свойства
   long&                       long_var          // переменная для получения информации
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

prop_id

[in] Идентификатор запрашиваемого свойства, значение перечисления ENUM_SERIES_INFO_INTEGER.

long_var

[out] Переменная, в которую помещается значение запрошенного свойства.

Возвращаемое значение

Значение типа long для первого варианта вызова.

Для второго варианта вызова возвращает true, если данное свойство поддерживается и значение было помещено в переменную long_var, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
void OnStart()
  {
//---
   Print("Количество баров по символу-периоду на данный момент = ",
         SeriesInfoInteger(Symbol(),Period(),SERIES_BARS_COUNT));

   Print("Самая первая дата по символу-периоду на данный момент = ",
```

© 2000-2025, MetaQuotes Ltd.

```
      (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_FIRSTDATE));

  Print("Самая первая дата в истории по символу на сервере = ",
      (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_SERVER_FIRSTDATE));

  Print("Данные по символу синхронизированы = ",
      (bool)SeriesInfoInteger(Symbol(),Period(),SERIES_SYNCHRONIZED));
}
```

© 2000-2025, MetaQuotes Ltd.

Bars

Возвращает количество баров в истории по соответствующему символу периоду. Существует 2 варианта функции.

Запросить количество всех баров в истории

```
int  Bars(
   string           symbol_name,      // имя символа
   ENUM_TIMEFRAMES  timeframe         // периоду
   );
```

Запросить количество баров на заданном интервале

```
int  Bars(
   string           symbol_name,      // имя символа
   ENUM_TIMEFRAMES  timeframe,        // период
   datetime         start_time,       // с какой даты
   datetime         stop_time         // по какую дату
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

Возвращаемое значение

Если указаны параметры start_time и stop_time, то функция возвращает количество баров в диапазоне дат. Если эти параметры не указаны, то функция возвращает общее количество баров.

Примечание

Если данные для таймсерии с указанными параметрами при вызове функции Bars() еще не сформированы в терминале, или данные таймсерии в момент вызова функции не синхронизированы с торговым сервером, то функция вернет нулевое значение.

При запросе количества баров в заданном диапазоне дат учитываются только те бары, чье время открытия попадает в этот диапазон. Например, если текущий день недели — суббота, то при запросе количества недельных баров с указанием start_time=последний_вторник и stop_time=последняя_пятница функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, и ни один недельный бар не попадает в указанный диапазон.

© 2000-2025, MetaQuotes Ltd.

Пример запроса количества всех баров в истории:

```
int bars=Bars(_Symbol,_Period);
if(bars>0)
  {
   Print("Количество баров в истории терминала по символу-периоду на данный момент
  }
else  //нет доступных баров
  {
   //--- видимо, данные по символу не синхронизированы с данными на сервере
   bool synchronized=false;
   //--- счетчик цикла
   int attempts=0;
   // сделаем 5 попыток дождаться синхронизации
   while(attempts<5)
     {
      if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCHRONIZED))
        {
         //--- есть синхронизация, выходим
         synchronized=true;
         break;
        }
      //--- увеличим счетчик
      attempts++;
      //--- подождем 10 миллисекунд до следующей итерации
      Sleep(10);
     }
   //--- вышли из цикла по факту синхронизации
   if(synchronized)
     {
      Print("Количество баров в истории терминала по символу-периоду на данный моме
      Print("Самая первая в истории терминала дата по символу-периоду на данный мом
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
      Print("Самая первая дата в истории по символу на сервере = ",
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
     }
   //--- синхронизация данных так и не была достигнута
   else
     {
      Print("Не удалось получить количество баров на ",_Symbol);
     }
  }
```

Пример запроса количества баров в заданном интервале:

```
int n;
datetime date1 = D'2016.09.02 23:55'; // пятница
datetime date2 = D'2016.09.05 00:00'; // понедельник
datetime date3 = D'2016.09.08 00:00'; // четверг
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   n=Bars(_Symbol,PERIOD_H1,D'2016.09.02 02:05',D'2016.09.02 10:55');
   Print("Количество баров: ",n); // Выведет "Количество баров: 8", в подсчете будет у
   n=Bars(_Symbol,PERIOD_D1,date1,date2);
   Print("Количество баров: ",n); // Выведет "Количество баров: 1", поскольку в диапаз
   n=Bars(_Symbol,PERIOD_W1,date2,date3);
   Print("Количество баров: ",n); // Выведет "Количество баров: 0", поскольку в заданн
```

Смотри также

Функции обработки событий

© 2000-2025, MetaQuotes Ltd.

BarsCalculated

Возвращает количество рассчитанных данных для запрашиваемого индикатора.

```
int  BarsCalculated(
   int       indicator_handle,     // handle индикатора
   );
```

Параметры

indicator_handle

[in] Хэндл индикатора, полученный соответствующей индикаторной функцией.

Возвращаемое значение

Возвращает количество рассчитанных данных в индикаторном буфере или -1 в случае ошибки (данные еще не рассчитаны).

Примечание

Функция полезна в тех случаях, когда необходимо получить данные индикатора сразу после его создания (получения хэндла индикатора).

Пример:

```
void OnStart()
  {
   double Ups[];
//--- установим для массивов признак таймсерии
   ArraySetAsSeries(Ups,true);
//--- создадим хэндл индикатора Fractals
   int FractalsHandle=iFractals(NULL,0);
//--- сбросим код ошибки
   ResetLastError();
//--- попытаемся скопировать значения индикатора
   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("Не удалось скопировать верхние фракталы. Error = ",GetLastError(),
         "i=",i,"    copied= ",copied);
         return;
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      else
        Print("Удалось скопировать верхние фракталы.",
        " i = ",i,"    copied = ",copied);
    }
  else Print("Удалось скопировать верхние фракталы. ArraySize = ",ArraySize(Ups));
}
```

© 2000-2025, MetaQuotes Ltd.

IndicatorCreate

Возвращает хэндл указанного технического индикатора, созданного на основе массива параметров типа MqlParam.

```
int  IndicatorCreate(
   string           symbol,                    // имя символа
   ENUM_TIMEFRAMES  period,                    // период
   ENUM_INDICATOR   indicator_type,            // тип индикатора из перечислен
   int              parameters_cnt=0,          // количество параметров
   const MqlParam&  parameters_array[]=NULL,   // массив параметров
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

indicator_type

[in] Тип индикатора, может принимать одно из значений перечисления ENUM_INDICATOR.

parameters_cnt

[in] Количество параметров, передаваемых в массиве parameters_array[]. Элементы массива имеют специальный тип структуры MqlParam. По умолчанию нулевое значение - параметры не передаются. Если указано ненулевое количество параметров, то параметр *parameters_array* является обязательным. Можно передавать не более 64 параметров.

parameters_array[]=NULL

[in] Массив типа MqlParam, элементы которого содержат тип и значение каждого входного параметра технического индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE.

Примечание

Если создается хэндл индикатора типа IND_CUSTOM, то поле *type* первого элемента массива входных параметров *parameters_array* обязательно должен иметь значение TYPE_STRING из перечисления ENUM_DATATYPE, а поле *string_value* первого элемента должно содержать имя пользовательского индикатора. Пользовательский индикатор должен быть скомпилирован (файл с расширением EX5) и находиться в директории MQL5/Indicators клиентского терминала или вложенной поддиректории.

Необходимые для тестирования индикаторы определяются автоматически из вызова функций iCustom(), если соответствующий параметр задан константной строкой. Для остальных случаев

© 2000-2025, MetaQuotes Ltd.

(использование функции IndicatorCreate() или использование неконстантной строки в параметре, задающем имя индикатора) необходимо указать свойство #property tester_indicator:

```
#property tester_indicator "indicator_name.ex5"
```

Если в пользовательском индикаторе используется первая форма вызова, то при передаче входных параметров последним параметром можно дополнительно указать на каких данных он будет рассчитываться. Если параметр "Apply to" не указан явно, то по умолчанию расчет производится по значениям PRICE_CLOSE.

Пример:

```
void OnStart()
  {
   MqlParam params[];
   int      h_MA,h_MACD;
//--- create iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE);
   ArrayResize(params,4);
//--- set ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=8;
//--- set ma_shift
   params[1].type         =TYPE_INT;
   params[1].integer_value=0;
//--- set ma_method
   params[2].type         =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- set applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- create MA
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- create iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA);
   ArrayResize(params,4);
//--- set fast ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=12;
//--- set slow ma_period
   params[1].type         =TYPE_INT;
   params[1].integer_value=26;
//--- set smooth period for difference
   params[2].type         =TYPE_INT;
   params[2].integer_value=9;
//--- set indicator handle as applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=h_MA;
//--- create MACD based on moving average
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- use indicators
```

© 2000-2025, MetaQuotes Ltd.

```
//--- . . .
//--- release indicators (first h_MACD)
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

© 2000-2025, MetaQuotes Ltd.

IndicatorParameters

Возвращает по указанному хэндлу количество входных параметров индикатора, а также сами значения и тип параметров.

```
int  IndicatorParameters(
   int               indicator_handle,     // хэндл индикатора
   ENUM_INDICATOR&   indicator_type,       // переменная для получения типа индикатора
   MqlParam&         parameters[]          // массив для получения параметров
   );
```

Параметры

indicator_handle

[in] Хэндл индикатора, для которого необходимо узнать количество параметров, на которых он рассчитан.

indicator_type

[out] Переменная типа ENUM_INDICATOR, в которую будет записан тип индикатора.

parameters[]

[out] Динамический массив для получения значений типа MqlParam, в который будет записан список параметров индикатора. Размер массива возвращает сама функция IndicatorParameters().

Возвращаемое значение

Количество входных параметров индикатора с указанным хэндлом, в случае ошибки возвращает -1. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

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

//--- количество окон на графике (всегда есть хотя бы одно главное окно)
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
//--- проходим по окнам графика
   for(int w=0;w<windows;w++)
     {
      //--- сколько индикаторов в данном окне/подконе
      int total=ChartIndicatorsTotal(0,w);
      //--- переберем все индикаторы в окне
      for(int i=0;i<total;i++)
        {
         //--- получим короткое имя индикатора
         string name=ChartIndicatorName(0,w,i);
         //--- получим хэндл индикатора
         int handle=ChartIndicatorGet(0,w,name);
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- выведем в журнал
         PrintFormat("Window=%d,  indicator #%d,  handle=%d",w,i,handle);
         //---
         MqlParam parameters[];
         ENUM_INDICATOR indicator_type;
         int params=IndicatorParameters(handle,indicator_type,parameters);
         //--- заголовок сообщения
         string par_info="Short name "+name+", type "
                         +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
         //---
         for(int p=0;p<params;p++)
           {
            par_info+=StringFormat("parameter %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);
        }
      //--- прошли по всем индикаторам в окне
     }
//---
  }
```

Смотри также

ChartIndicatorGet()

© 2000-2025, MetaQuotes Ltd.

IndicatorRelease

Удаляет хэндл индикатора и освобождает расчетную часть индикатора, если ею больше никто не пользуется.

```
bool  IndicatorRelease(
   int       indicator_handle     // handle индикатора
   );
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Функция позволяет удалять хэндл индикатора, если он больше не нужен, и таким образом позволяет экономить память. Удаление хендла производится сразу, удаление расчетной части индикатора производится через некоторое небольшое время (если обращений к ней больше нет).

При работе в тестере стратегий функция IndicatorRelease() не выполняется.

Пример:

```
//+------------------------------------------------------------------+
//|                                      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"
//--- input parameters
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;
//--- будем хранить хендл индикатора
int MA_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- создадим хендл индикатора
   MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,PRICE_CLOSE);
//--- удаление глобальной переменной
   if(GlobalVariableCheck("MA_value"))
      GlobalVariableDel("MA_value");
//---
   return(INIT_SUCCEEDED);
  }
```

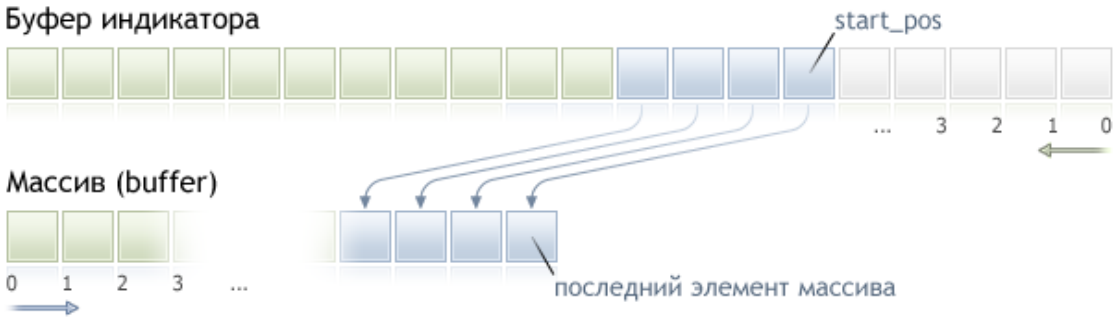
© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- если глобальной переменной еще нет
   if(!GlobalVariableCheck("MA_value"))
     {
      //--- получим значения индикатора на двух последних барах
      if(MA_handle!=INVALID_HANDLE)
        {
         //--- динамический массив для получения значений индикатора
         double values[];
         if(CopyBuffer(MA_handle,0,0,2,values)==2 && values[0]!=EMPTY_VALUE)
           {
            //--- запомним в глобальной переменной значение на предпоследнем баре
            if(GlobalVariableSet("MA_value",values[0]))
              {
               //--- освободим хэндл индикатора
               if(!IndicatorRelease(MA_handle))
                  Print("IndicatorRelease() failed. Error ",GetLastError());
               else MA_handle=INVALID_HANDLE;
              }
            else
               Print("GlobalVariableSet failed. Error ",GetLastError());
           }
        }
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

CopyBuffer

Получает в массив buffer данные указанного буфера указанного индикатора в указанном количестве.



Отсчет элементов копируемых данных (индикаторный буфер с индексом buffer_num) от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар (значение индикатора для текущего бара).

При копировании заранее неизвестного количества данных в качестве массива-приемника buffer[] желательно использовать динамический массив, так как функция CopyBuffer() старается распределить размер принимающего массива под размер копируемых данных. Если в качестве принимающего массива buffer[] выступает индикаторный буфер (массив, предварительно назначенный под хранение значений индикатора функцией SetIndexBufer()), то допускается частичное копирование. Пример можно посмотреть в пользовательском индикаторе Awesome_Oscillator.mq5 из стандартной поставки терминала.

Если необходимо произвести частичное копирование значений индикатора в другой массив (не индикаторный буфер), то для этих целей необходимо использовать промежуточный массив, в который копируется требуемое количество. И уже из этого массива-посредника произвести поэлементное копирование нужного количества значений в нужные места принимающего массива.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopyBuffer(
   int       indicator_handle,     // handle индикатора
   int       buffer_num,           // номер буфера индикатора
   int       start_pos,            // откуда начнем
   int       count,                // сколько копируем
   double    buffer[]              // массив, куда будут скопированы данные
   );
```

Обращение по начальной дате и количеству требуемых элементов

© 2000-2025, MetaQuotes Ltd.

```
int  CopyBuffer(
   int       indicator_handle,     // handle индикатора
   int       buffer_num,           // номер буфера индикатора
   datetime  start_time,           // с какой даты
   int       count,                // сколько копируем
   double    buffer[]              // массив, куда будут скопированы данные
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

```
int  CopyBuffer(
   int       indicator_handle,     // handle индикатора
   int       buffer_num,           // номер буфера индикатора
   datetime  start_time,           // с какой даты
   datetime  stop_time,            // по какую дату
   double    buffer[]              // массив, куда будут скопированы данные
   );
```

Параметры

indicator_handle

[in] Хэндл индикатора, полученный соответствующей индикаторной функцией.

buffer_num

[in] Номер индикаторного буфера.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

buffer[]

[out] Массив типа double.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, то функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если

© 2000-2025, MetaQuotes Ltd.

данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута.

Пример:

```
//+------------------------------------------------------------------+
//|                                                TestCopyBuffer3.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"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
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;
//--- indicator buffers
double                   MABuffer[];
int                      ma_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Параметр AsSeries = ",AsSeries);
   Print("Индикаторный буфер после SetIndexBuffer() является таймсерией = ",
         ArrayGetAsSeries(MABuffer));
//--- set short indicator name
   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- set AsSeries (depends on input parameter)
   ArraySetAsSeries(MABuffer,AsSeries);
   Print("Индикаторный буфер после ArraySetAsSeries(MABuffer,true); является таймсерие
         ArrayGetAsSeries(MABuffer));
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- check if all data calculated
   if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- we can copy not all data
   int to_copy;
   if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
   else
     {
      to_copy=rates_total-prev_calculated;
      //--- last value is always copied
      to_copy++;
     }
//--- try to copy
   if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

В вышеприведенном примере проиллюстрировано заполнение индикаторного буфера значениями другого индикаторного буфера от индикатора на том же символе/периоде.

Более полный пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

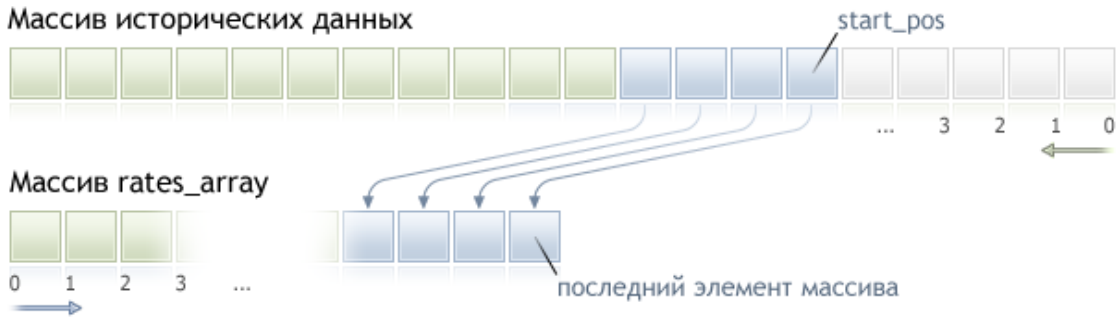
Смотри также

Свойства пользовательских индикаторов, SetIndexBuffer

© 2000-2025, MetaQuotes Ltd.

CopyRates

Получает в массив rates_array исторические данные структуры MqlRates указанного символа-периода в указанном количестве. Отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемного массива использовать динамический массив, так как если данных оказывается меньше (или больше), чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopyRates(
   string           symbol_name,       // имя символа
   ENUM_TIMEFRAMES  timeframe,         // период
   int              start_pos,         // откуда начнем
   int              count,             // сколько копируем
   MqlRates         rates_array[]      // массив, куда будут скопированы данные
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
int  CopyRates(
   string           symbol_name,       // имя символа
   ENUM_TIMEFRAMES  timeframe,         // период
   datetime         start_time,        // с какой даты
   int              count,             // сколько копируем
   MqlRates         rates_array[]      // массив, куда будут скопированы данные
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

```
int  CopyRates(
```

© 2000-2025, MetaQuotes Ltd.

```
   string           symbol_name,       // имя символа
   ENUM_TIMEFRAMES  timeframe,         // период
   datetime         start_time,        // с какой даты
   datetime         stop_time,         // по какую дату
   MqlRates         rates_array[]      // массив, куда будут скопированы данные
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_time

[in] Время бара, соответствующее первому элементу.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

stop_time

[in] Время бара, соответствующее последнему элементу.

rates_array[]

[out] Массив типа MqlRates.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, то функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и учитывается с точностью до секунды. То есть дата открытия любого бара, для которого

© 2000-2025, MetaQuotes Ltd.

возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

Пример:

```
void OnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Скопировано баров: "+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("Не удалось получить исторические данные по символу ",Symbol());
  }
```

Более полный пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и

© 2000-2025, MetaQuotes Ltd.

вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

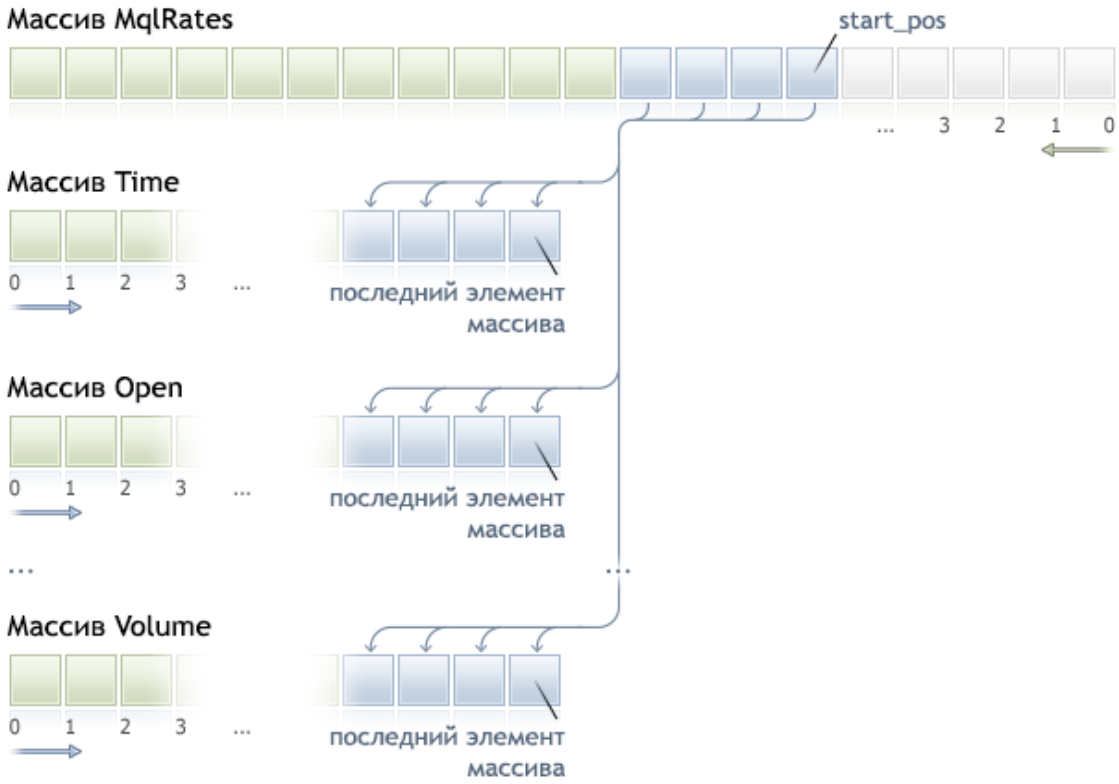
Смотри также

Структуры и классы, TimeToString, StringFormat

© 2000-2025, MetaQuotes Ltd.

CopySeries

Получает в указанный набор массивов синхронизированные таймсерии из структуры MqlRates для указанного символа-периода в указанном количестве. Отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемных массивов использовать динамический массив, так как если данных оказывается больше, чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент таймсерии будет скопирован в начало физической памяти, отведенной под массив.

```
int  CopySeries(
   string           symbol_name,       // имя символа
   ENUM_TIMEFRAMES  timeframe,         // период
   int              start_pos,         // откуда начнем
   int              count,             // сколько копируем
   ulong            rates_mask,        // комбинация флагов для указания запрашиваемых
   void&            array1[],          // массив, куда будут скопированы данные первой
   void&            array2[]           // массив, куда будут скопированы данные второй
   ...
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

rates_mask

[in] Комбинация флагов из перечисления ENUM_COPY_RATES .

array1, array2,...

[out] Массив соответствующего типа для получения таймсерии из структуры MqlRates. Порядок массивов, передаваемых в функцию, должен соответствовать порядку полей в структуре MqlRates.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых таймсерий полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае если запрашиваемые данные находятся за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, то функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

Разница между CopySeries и CopyRates

Функция CopySeries позволяет за один раз получать только нужные таймсерии в разные указанные массивы, при этом они все синхронизированы между собой. Это означает, что все значения в полученных массивах по конкретному индексу N будут принадлежать одному и тому же бару на указанной паре Символ/Таймрейм. В этом случае программисту не требуется заботиться о том, чтобы все полученные таймсерии были синхронизированы по времени открытия бара.

© 2000-2025, MetaQuotes Ltd.

В отличие от CopyRates, которая возвращает полный набор таймсерий в виде массива MqlRates, функция CopySeries позволяет программисту получать только нужные таймсерии в виде отдельных массивов, для этого необходимо указать комбинацию флагов для указания типа запрашиваемых таймсерий. При этом порядок массивов передаваемых в функцию должен соответствовать порядку полей в структуре MqlRates:

```
struct MqlRates
  {
   datetime time;         // время начала периода
   double   open;         // цена открытия
   double   high;         // наивысшая цена за период
   double   low;          // наименьшая цена за период
   double   close;        // цена закрытия
   long     tick_volume;  // тиковый объем
   int      spread;       // спред
   long     real_volume;  // биржевой объем
  }
```

Таким образом, если необходимо получить значения таймсерий *time*, *close* и *real_volume* для последних 100 баров текущего Символа/Таймфрема, то вызов должен быть следующим:

```
datetime time[];
double   close[];
long     volume[];
CopySeries(NULL,0,0,100,COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_VOLUME_REAL,time,c
```

При этом важен порядок массивов "*time*, *close*, *volume*" — он должен соответствовать порядку полей в структуре MqlRates. А вот порядок значений в маске *rates_mask* значения не имеет, маска могла быть такой :

```
COPY_RATES_VOLUME_REAL|COPY_RATES_TIME|COPY_RATES_CLOSE
```

Пример:

```
//--- 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)
  {
//--- массивы для получения таймсерий из ценовой структуры MqlRates
   double   open[];
   double   close[];
   float    closef[];
   datetime time1[], time2[];
//---запросим цены закрытия в массив типа double
   ResetLastError();
```

© 2000-2025, 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);

//--- теперь запросим еще цены открытия, а цены закрытия в массив типа float
   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);
//--- сравним полученные данные
   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]);
        }
     }
//--- Результат
      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   |
//---
  }
```

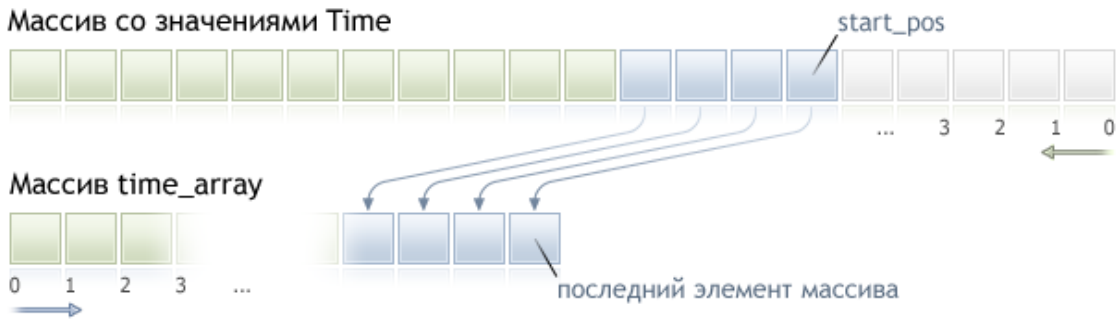
Смотри также

Структуры и классы, CopyRates

© 2000-2025, MetaQuotes Ltd.

CopyTime

Функция получает в массив time_array исторические данные времени открытия баров для указанной пары символ-период в указанном количестве. Необходимо отметить, что отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемного массива использовать динамический массив, так как если данных оказывается меньше (или больше), чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopyTime(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   int               start_pos,        // откуда начнем
   int               count,            // сколько копируем
   datetime          time_array[]      // массив для копирования времени открытия
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
int  CopyTime(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   datetime          start_time,       // с какой даты
   int               count,            // сколько копируем
   datetime          time_array[]      // массив для копирования времени открытия
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

© 2000-2025, MetaQuotes Ltd.

```
int  CopyTime(
   string           symbol_name,      // имя символа
   ENUM_TIMEFRAMES  timeframe,        // период
   datetime         start_time,       // с какой даты
   datetime         stop_time,        // по какую дату
   datetime         time_array[]      // массив для копирования времени открытия
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

time_array[]

[out] Массив типа datetime.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

© 2000-2025, MetaQuotes Ltd.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и учитывается с точностью до секунды. То есть дата открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

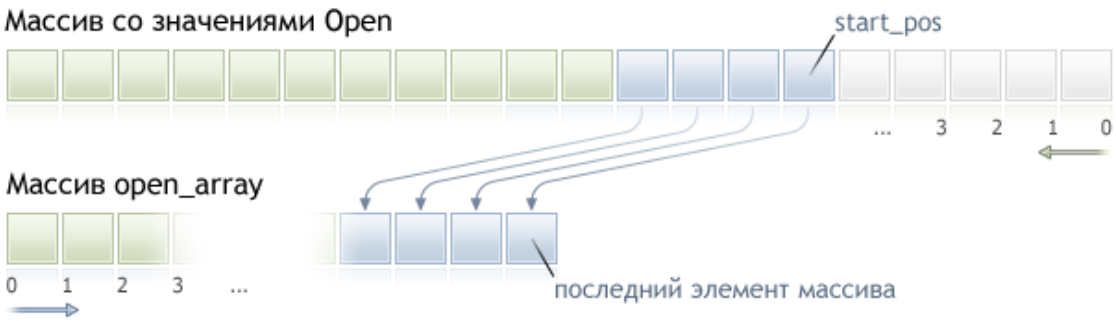
Пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyOpen

Функция получает в массив open_array исторические данные цен открытия баров для указанной пары символ-период в указанном количестве. Необходимо отметить, что отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемного массива использовать динамический массив, так как если данных оказывается меньше (или больше), чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приёмный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopyOpen(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   int               start_pos,        // откуда начнем
   int               count,            // сколько копируем
   double            open_array[]      // массив для копирования цен открытия
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
int  CopyOpen(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   datetime          start_time,       // с какой даты
   int               count,            // сколько копируем
   double            open_array[]      // массив для копирования цен открытия
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

© 2000-2025, MetaQuotes Ltd.

```
int  CopyOpen(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   datetime          start_time,       // с какой даты
   datetime          stop_time,        // по какую дату
   double            open_array[]      // массив для копирования цен открытия
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

open_array[]

[out] Массив типа double.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и

© 2000-2025, MetaQuotes Ltd.

учитывается с точностью до секунды. То есть дата открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

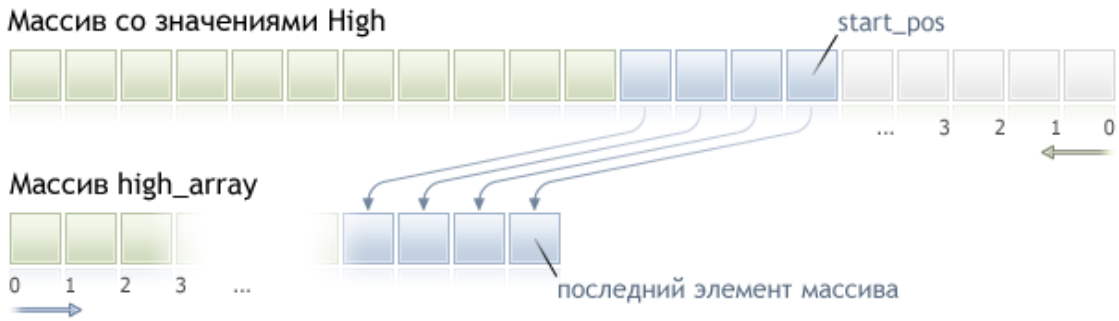
Пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyHigh

Функция получает в массив high_array исторические данные максимальных цен баров для указанной пары символ-период в указанном количестве. Необходимо отметить, что отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемного массива использовать динамический массив, так как если данных оказывается меньше (или больше), чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopyHigh(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   int               start_pos,        // откуда начнем
   int               count,            // сколько копируем
   double            high_array[]      // массив для копирования максимальных цен
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
int  CopyHigh(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   datetime          start_time,       // с какой даты
   int               count,            // сколько копируем
   double            high_array[]      // массив для копирования максимальных цен
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

© 2000-2025, MetaQuotes Ltd.

```
int  CopyHigh(
   string           symbol_name,      // имя символа
   ENUM_TIMEFRAMES  timeframe,        // период
   datetime         start_time,       // с какой даты
   datetime         stop_time,        // по какую дату
   double           high_array[]      // массив для копирования максимальных цен
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

high_array[]

[out] Массив типа double.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае, если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), то функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, то функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться и при следующем аналогичном запросе функция вернет уже больше данных.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и

© 2000-2025, MetaQuotes Ltd.

учитывается с точностью до секунды. То есть дата открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть, время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

Пример:

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Пример вывода значений High[i] и Low[i]"
#property description "для баров, выбранных случайным образом"

double High[],Low[];
//+------------------------------------------------------------------+
//| Получим Low для заданного номера бара                            |
//+------------------------------------------------------------------+
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);
  }
//+------------------------------------------------------------------+
//| Получим High для заданного номера бара                           |
//+------------------------------------------------------------------+
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);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- выводим на каждом тике значения High и Low для бара с индексом,
//--- равным секунде поступления тика
   datetime t=TimeCurrent();
   int sec=t%60;
   printf("High[%d] = %G  Low[%d] = %G",
          sec,iHigh(Symbol(),0,sec),
          sec,iLow(Symbol(),0,sec));
  }
```

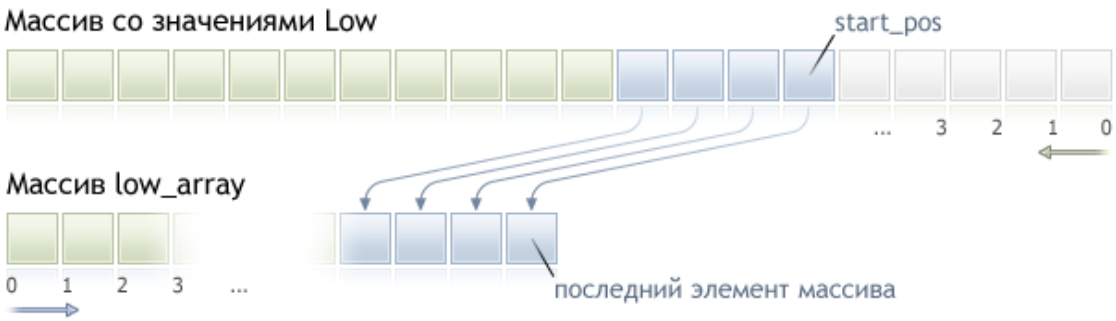
Более полный пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyLow

Функция получает в массив low_array исторические данные минимальных цен баров для указанной пары символ-период в указанном количестве. Необходимо отметить, что отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемного массива использовать динамический массив, так как если данных оказывается меньше (или больше), чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopyLow(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   int               start_pos,        // откуда начнем
   int               count,            // сколько копируем
   double            low_array[]       // массив для копирования минимальных цен
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
int  CopyLow(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   datetime          start_time,       // с какой даты
   int               count,            // сколько копируем
   double            low_array[]       // массив для копирования минимальных цен
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

© 2000-2025, MetaQuotes Ltd.

```
int  CopyLow(
   string           symbol_name,      // имя символа
   ENUM_TIMEFRAMES  timeframe,        // период
   datetime         start_time,       // с какой даты
   datetime         stop_time,        // по какую дату
   double           low_array[]       // массив для копирования минимальных цен
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

low_array[]

[out] Массив типа double.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и

© 2000-2025, MetaQuotes Ltd.

учитывается с точностью до секунды. То есть дата открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

Пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

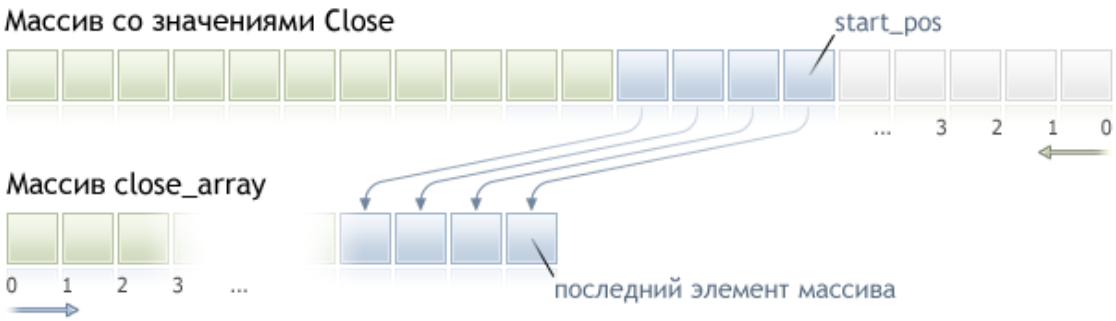
Смотри также

CopyHigh

© 2000-2025, MetaQuotes Ltd.

CopyClose

Функция получает в массив close_array исторические данные цен закрытия баров для указанной пары символ-период в указанном количестве. Необходимо отметить, что отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемного массива использовать динамический массив, так как если данных оказывается меньше (или больше), чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopyClose(
   string           symbol_name,       // имя символа
   ENUM_TIMEFRAMES  timeframe,         // период
   int              start_pos,         // откуда начнем
   int              count,             // сколько копируем
   double           close_array[]      // массив для копирования цен закрытия
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
int  CopyClose(
   string           symbol_name,       // имя символа
   ENUM_TIMEFRAMES  timeframe,         // период
   datetime         start_time,        // с какой даты
   int              count,             // сколько копируем
   double           close_array[]      // массив для копирования цен закрытия
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

© 2000-2025, MetaQuotes Ltd.

```
int  CopyClose(
   string           symbol_name,       // имя символа
   ENUM_TIMEFRAMES  timeframe,         // период
   datetime         start_time,        // с какой даты
   datetime         stop_time,         // по какую дату
   double           close_array[]      // массив для копирования цен закрытия
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

close_array[]

[out] Массив типа double.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и

© 2000-2025, MetaQuotes Ltd.

учитывается с точностью до секунды. То есть дата открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

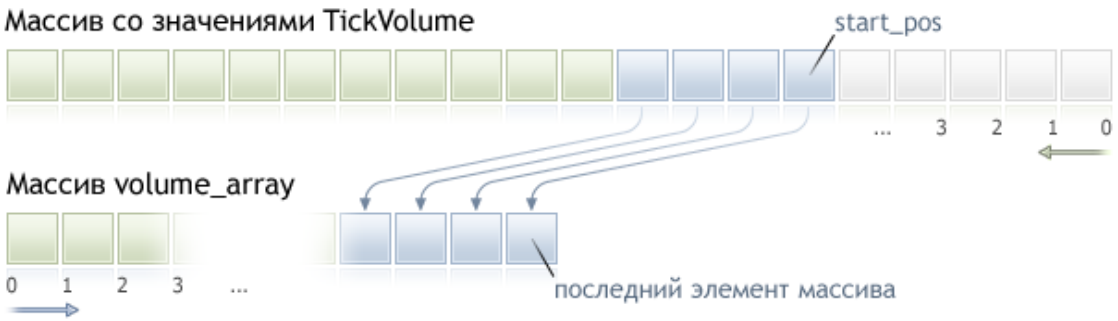
Пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyTickVolume

Функция получает в массив volume_array исторические данные тиковых объемов для указанной пары символ-период в указанном количестве. Необходимо отметить, что отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемного массива использовать динамический массив, так как если данных оказывается меньше (или больше), чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopyTickVolume(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   int               start_pos,        // откуда начнем
   int               count,            // сколько копируем
   long              volume_array[]    // массив для копирования тиковых объемов
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
int  CopyTickVolume(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   datetime          start_time,       // с какой даты
   int               count,            // сколько копируем
   long              volume_array[]    // массив для копирования тиковых объемов
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

© 2000-2025, MetaQuotes Ltd.

```
int  CopyTickVolume(
   string           symbol_name,      // имя символа
   ENUM_TIMEFRAMES  timeframe,        // период
   datetime         start_time,       // с какой даты
   datetime         stop_time,        // по какую дату
   long             volume_array[]    // массив для копирования тиковых объемов
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

volume_array[]

[out] Массив типа long.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и

© 2000-2025, MetaQuotes Ltd.

учитывается с точностью до секунды. То есть дата открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

Пример:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot 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
//--- input parameters
input int      bars=3000;
//--- indicator buffers
double         TickVolumeBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                  const double &high[],
                  const double &low[],
                  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("Получено исторических значений TickVolume: "+prices);
     }
   else
     {
      long timeseries[];
      int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
      TickVolumeBuffer[rates_total-1]=timeseries[0];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

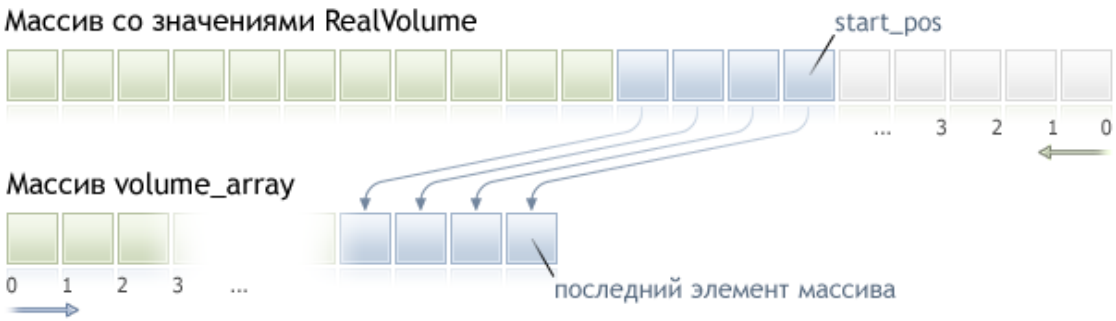
Более полный пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyRealVolume

Функция получает в массив volume_array исторические данные торговых объемов для указанной пары символ-период в указанном количестве. Необходимо отметить, что отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемного массива использовать динамический массив, так как если данных оказывается меньше (или больше), чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopyRealVolume(
   string           symbol_name,      // имя символа
   ENUM_TIMEFRAMES  timeframe,        // период
   int              start_pos,        // откуда начнем
   int              count,            // сколько копируем
   long             volume_array[]    // массив для копирования объемов
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
int  CopyRealVolume(
   string           symbol_name,      // имя символа
   ENUM_TIMEFRAMES  timeframe,        // период
   datetime         start_time,       // с какой даты
   int              count,            // сколько копируем
   long             volume_array[]    // массив для копирования объемов
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

© 2000-2025, MetaQuotes Ltd.

```
int  CopyRealVolume(
   string           symbol_name,      // имя символа
   ENUM_TIMEFRAMES  timeframe,        // период
   datetime         start_time,       // с какой даты
   datetime         stop_time,        // по какую дату
   long             volume_array[]    // массив для копирования объемов
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

volume_array[]

[out] Массив типа long.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и

© 2000-2025, MetaQuotes Ltd.

учитывается с точностью до секунды. То есть дата открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

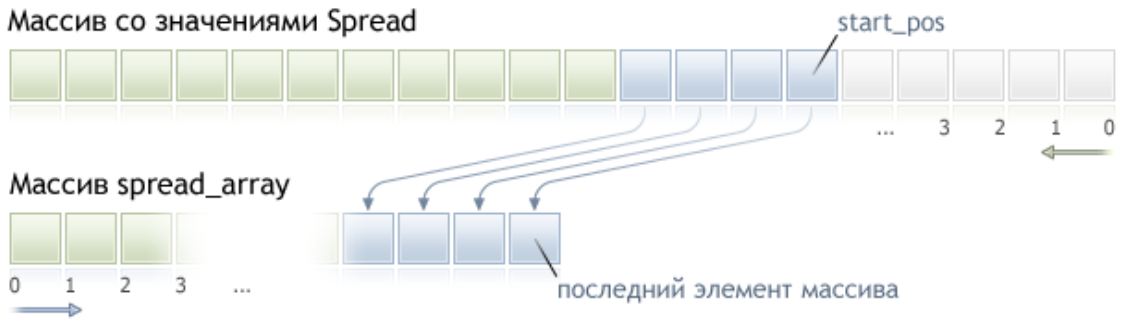
Более полный пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopySpread

Функция получает в массив spread_array исторические данные спредов для указанной пары символ-период в указанном количестве. Необходимо отметить, что отсчет элементов от стартовой позиции ведется от настоящего к прошлому, то есть стартовая позиция, равная 0, означает текущий бар.



При копировании заранее неизвестного количества данных рекомендуется в качестве приемного массива использовать динамический массив, так как если данных оказывается меньше (или больше), чем вмещает массив, то производится попытка перераспределения массива таким образом, чтобы запрошенные данные поместились целиком и полностью.

Если необходимо копировать заранее известное количество данных, то лучше это делать в статически выделенный буфер, чтобы избежать излишнего перевыделения памяти.

Неважно, какое свойство имеет приемный массив - as_series=true или as_series=false, данные будут скопированы таким образом, что самый старый по времени элемент будет в начале физической памяти, отведенной под массив. Существует 3 варианта функции.

Обращение по начальной позиции и количеству требуемых элементов

```
int  CopySpread(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   int               start_pos,        // откуда начнем
   int               count,            // сколько копируем
   int               spread_array[]    // массив для копирования спредов
   );
```

Обращение по начальной дате и количеству требуемых элементов

```
int  CopySpread(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   datetime          start_time,       // с какой даты
   int               count,            // сколько копируем
   int               spread_array[]    // массив для копирования спредов
   );
```

Обращение по начальной и конечной датам требуемого интервала времени

© 2000-2025, MetaQuotes Ltd.

```
int  CopySpread(
   string            symbol_name,      // имя символа
   ENUM_TIMEFRAMES   timeframe,        // период
   datetime          start_time,       // с какой даты
   datetime          stop_time,        // по какую дату
   int               spread_array[]    // массив для копирования спредов
   );
```

Параметры

symbol_name

[in] Символ.

timeframe

[in] Период.

start_pos

[in] Номер первого копируемого элемента.

count

[in] Количество копируемых элементов.

start_time

[in] Время бара, соответствующее первому элементу.

stop_time

[in] Время бара, соответствующее последнему элементу.

spread_array[]

[out] Массив типа int.

Возвращаемое значение

Количество скопированных элементов массива либо -1 в случае ошибки.

Примечание

Если интервал запрашиваемых данных полностью находится вне доступных данных на сервере, то функция возвращает -1. В случае если запрашиваются данные за пределами TERMINAL_MAXBARS (максимальное количество баров на графике), функция также вернет -1.

При запросе данных из индикатора, если запрашиваемые таймсерии еще не построены или их необходимо загрузить с сервера, функция сразу же вернет -1, но при этом сам процесс загрузки/построения будет инициирован.

При запросе данных из эксперта или скрипта, будет инициирована загрузка с сервера, если локально этих данных у терминала нет, либо начнется построение нужной таймсерии, если данные можно построить из локальной истории, но они еще не готовы. Функция вернет то количество данных, которые будут готовы к моменту истечения таймаута, но загрузка истории будет продолжаться, и при следующем аналогичном запросе функция вернет уже больше данных.

При запросе данных по начальной дате и количеству требуемых элементов возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и

© 2000-2025, MetaQuotes Ltd.

учитывается с точностью до секунды. То есть дата открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда равна или меньше указанной.

При запросе данных в заданном диапазоне дат возвращаются только данные, попадающие в запрашиваемый интервал, при этом интервал задается и учитывается с точностью до секунды. То есть время открытия любого бара, для которого возвращается значение (объем, спред, значение в индикаторном буфере, цена Open, High, Low, Close или время открытия Time), всегда находится в запрошенном интервале.

Таким образом, если текущий день недели Суббота, то при попытке скопировать данные на недельном таймфрейме с указанием *start_time=Последний_Вторник* и *stop_time=Последняя_Пятница* функция вернет 0, так как время открытия на недельном таймфрейме всегда приходится на воскресенье, но ни один недельный бар не попадает в указанный диапазон.

Если необходимо получить значение, соответствующее текущему незавершенному бару, то можно использовать первую форму вызова с указанием *start_pos*=0 и *count*=1.

Пример:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Spread
#property indicator_label1  "Spread"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      bars=3000;
//--- indicator buffers
double         SpreadBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                  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("Получено исторических значений спреда: ",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();
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Более полный пример запроса исторических данных смотрите в разделе Способы привязки объектов. В приведенном там скрипте показано, как получать значения индикатора iFractals на последних 1000 барах и как потом вывести на график по десять последних фракталов вверх и вниз. Подобный прием можно использовать для всех индикаторов, которые имеют пропуски значений и обычно реализуются с помощью следующих стилей построения:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyTicks

Функция получает в массив ticks_array тики в формате MqlTick, при этом индексация ведётся от прошлого к настоящему, то есть тик с индексом 0 является самым старым в массиве. Для анализа тика необходимо проверять поле *flags*, которое сообщает, что именно было изменено в данном тике.

```
int  CopyTicks(
   string           symbol_name,           // имя символа
   MqlTick&         ticks_array[],         // массив для приёма тиков
   uint             flags=COPY_TICKS_ALL,  // флаг, определяющий тип получаемых тиков
   ulong            from=0,                // дата, начиная с которой запрашиваются ти
   uint             count=0                // количество тиков, которые необходимо пол
   );
```

Параметры

symbol_name

[in] Символ.

ticks_array

[out] Массив типа MqlTick для приема тиков.

flags

[in] Флаг, определяющий тип запрашиваемых тиков. COPY_TICKS_INFO - тики, вызванные изменениями Bid и/или Ask, COPY_TICKS_TRADE - тики с изменения Last и Volume, COPY_TICKS_ALL - все тики. При любом типе запроса в оставшиеся поля структуры MqlTick дописываются значения предыдущего тика.

from

[in] Дата, начиная с которой запрашиваются тики. Указывается в миллисекундах с 01.01.1970. Если параметр *from*=0, то отдаются последние *count* тиков.

count

[in] Количество запрашиваемых тиков. Если параметры from и count не указаны, то в массив ticks_array[] будут записаны все доступные последние тики, но не более 2000.

Возвращаемое значение

Количество скопированных тиков либо -1 в случае ошибки.

Примечание

Функция CopyTicks() позволяет запрашивать и анализировать все пришедшие тики. Первый вызов CopyTicks() инициирует синхронизацию базы тиков, хранящихся на жёстком диске по данному символу. Если тиков в локальной базе не хватает, то недостающие тики автоматически будут загружены с торгового сервера. При этом будут синхронизированы тики с даты ***from***, указанной в CopyTicks(), по текущий момент. После этого все приходящие по данному символу тики будут поступать в тиковую базу и поддерживать её в актуальном синхронизированном состоянии.

Если параметры *from* и *count* не указаны, то в массив *ticks_array[]* будут записаны все доступные тики, но не более 2000. Параметр *flags* позволяет задать тип требуемых тиков.

© 2000-2025, MetaQuotes Ltd.

COPY_TICKS_INFO – отдаются тики, в которых есть изменения цены Bid и/или Ask. Но при этом будут также заполнены данные остальных полей, например, если изменилась только цена Bid, то в поля *ask* и *volume* будут записаны последние известные значения. Чтобы узнать точно, что именно изменилось, необходимо анализировать поле *flags*, которое будет иметь значение TICK_FLAG_BID и/или TICK_FLAG_ASK. Если тик имеет нулевые значения цен Bid и Ask, и при этом флаги показывают, что эти данные цены изменились (*flags*=TICK_FLAG_BID|TICK_FLAG_ASK), то это говорит об опустошении стакана заявок. Другими словами, в этот момент отсутствуют заявки на покупку и продажи.

COPY_TICKS_TRADE – отдаются тики, в которых есть изменения последней цены сделки и объема. Но при этом будут также заполнены данные остальных полей, то есть в поля Bid и Ask будут записаны последние известные значения. Чтобы узнать точно, что именно изменилось, необходимо анализировать поле *flags*, которое будет иметь значение TICK_FLAG_LAST и TICK_FLAG_VOLUME.

COPY_TICKS_ALL – отдаются все тики, в которых есть хоть какое-то изменение. При этом неизмененные поля также заполняются последними известными значениями.

Вызов CopyTicks() с флагом COPY_TICKS_ALL выдает сразу все тики из запрашиваемого диапазона, в то время как вызов в других режимах требует некоторого времени на предобработку и выборку тиков, и поэтому не даёт существенного выигрыша по скорости выполнения.

При запросе тиков (неважно, COPY_TICKS_INFO или COPY_TICKS_TRADE), в каждом тике содержится полная ценовая информация на момент тика (*bid, ask, last* и *volume*). Это сделано для удобства анализа торговой обстановки на момент каждого тика, чтобы не приходилось каждый раз запрашивать глубокую тиковую историю и искать в ней значения по другим полям.

В индикаторах функция CopyTicks() возвращает результат немедленно: При вызове из индикатора CopyTick() сразу же вернёт доступные по символу тики, а также запустит синхронизацию базы тиков, если данных не хватило. Все индикаторы на одном символе работают в одном общем потоке, поэтому индикатор не имеет права ждать завершения синхронизации. После окончания синхронизации при последующем вызове CopyTicks() вернёт все запрашиваемые тики. Функция OnCalculate() в индикаторах вызывается после поступления каждого тика.

В экспертах и скриптах функция CopyTicks() может дожидаться результата до 45 секунд: В отличие от индикатора каждый эксперт и скрипт работает в собственном потоке, и поэтому может дожидаться окончания синхронизации до 45 секунд. Если за это время тики так и не будут синхронизированы в необходимом объёме, то CopyTicks() по таймауту вернёт только имеющиеся в наличии тики, при этом синхронизация продолжится. Функция OnTick() в экспертах не является обработчиком каждого тика, она лишь уведомляет эксперта об изменениях на рынке. Изменения могут быть пакетными: в терминал может одновременно прийти несколько тиков, но функция OnTick() будет вызвана лишь один раз для уведомления эксперта о последнем состоянии рынка.

Скорость выдачи: терминал хранит по каждому символу 4096 последних тиков в кеше для быстрого доступа (для символов с запущенным стаканом – 65536 тиков), запросы к этим данным выполняются быстрее всего. При запросе тиков текущей торговой сессии за пределами кеша CopyTicks() обращается уже к тикам, которые хранятся в памяти терминала, эти запросы требуют большего времени на выполнение. Самыми медленными являются запросы тиков за другие дни, так как в этом случае данные читаются уже с диска.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- запрашиваем 100 миллионов тиков, чтобы гарантировать получение всей тиковой исто
input int      getticks=100000000; // сколько тиков требуется
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int     attempts=0;     // счетчик попыток
   bool    success=false;  // флаг успешного выполнения копирования тиков
   MqlTick tick_array[];   // массив для приема тиков
   MqlTick lasttick;       // для получения данных последнего тика
   SymbolInfoTick(_Symbol,lasttick);
//--- сделаем 3 попытки получить тики
   while(attempts<3)
     {
      //--- замерим время старта перед получением тиков
      uint start=GetTickCount();
//--- запросим тиковую историю с момента 1970.01.01 00:00.001 (параметр from=1 ms)
      int received=CopyTicks(_Symbol,tick_array,COPY_TICKS_ALL,1,getticks);
      if(received!=-1)
        {
         //--- выведем информацию о количестве тиков и затраченном времени времени
         PrintFormat("%s: received %d ticks in %d ms",_Symbol,received,GetTickCount()-
         //--- если тиковая история синхронизирована, то код ошибки равен нулю
         if(GetLastError()==0)
           {
            success=true;
            break;
           }
         else
            PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d
            _Symbol,received,GetTickCount()-start,_LastError);
        }
      //--- считаем попытки
      attempts++;
      //--- пауза в 1 секунду в ожидании завершения синхронизации тиковой базы
      Sleep(1000);
     }
//--- не удалось получить запрошенные тики от самого начала истории с трех попыток
   if(!success)
     {
      PrintFormat("Ошибка! Не удалось получить %d тиков по %s с трех попыток",getticks
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   int ticks=ArraySize(tick_array);
//--- выведем время первого тика в массиве
   datetime firstticktime=tick_array[ticks-1].time;
   PrintFormat("Last tick time = %s.%03I64u",
               TimeToString(firstticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar
//--- выведем время последнего тика в массиве
   datetime lastticktime=tick_array[0].time;
   PrintFormat("First tick time = %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);
//--- теперь выведем первые 100 тиков последнего дня
   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]));
        }
     }
//--- выведем первые 100 сделок последнего дня
   counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      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]));
```

© 2000-2025, MetaQuotes Ltd.

```
            }
         }
      }
   }
//+------------------------------------------------------------------+
//| возвращает строковое описание тика                               |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
   {
    string desc=StringFormat("%s.%03d ",
                             TimeToString(tick.time),tick.time_msc%1000);
//--- проверим флаги
    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);
//--- проверим сначала тик на торговые флаги
    if(buy_tick || sell_tick)
      {
       //--- сформируем вывод для торгового тика
       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
      {
       //--- для инфо тика сформируем вывод немного иначе
       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)";
      }
//--- вернем описание тика
    return desc;
   }
//+------------------------------------------------------------------+
/* Пример вывода
Si-12.16: received 11048387 ticks in 4937 ms
Last tick time = 2016.09.26 18:32:59.775
First tick time = 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)
```

© 2000-2025, MetaQuotes Ltd.

```
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)
*/
```

Смотри также

SymbolInfoTick, Структура для получения текущих цен, OnTick()

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange

Функция получает в массив ticks_array тики в формате MqlTick в указанном диапазоне дат. При этом индексация ведётся от прошлого к настоящему, то есть тик с индексом 0 является самым старым в массиве. Для анализа тика необходимо проверять поле *flags*, которое сообщает о том, что именно изменилось.

```
int  CopyTicksRange(
   const string      symbol_name,           // имя символа
   MqlTick&          ticks_array[],         // массив для приёма тиков
   uint              flags=COPY_TICKS_ALL,  // флаг, определяющий тип получаемых тиков
   ulong             from_msc=0,            // дата, начиная с которой запрашиваются ти
   ulong             to_msc=0               // дата, по которую запрашиваются тики
   );
```

Параметры

symbol_name

[in] Символ.

ticks_array

[out] Статический или динамический массив MqlTick для приема тиков. Если в статический массив не вмещаются все тики из запрошенного интервала времени, то будет получено столько тиков, сколько помещается в массив. При этом функция сгенерирует ошибку ERR_HISTORY_SMALL_BUFFER (4407) .

flags

[in] Флаг, определяющий тип запрашиваемых тиков. COPY_TICKS_INFO - тики, вызванные изменениями Bid и/или Ask, COPY_TICKS_TRADE - тики с изменения Last и Volume, COPY_TICKS_ALL - все тики. При любом типе запроса в оставшиеся поля структуры MqlTick дописываются значения предыдущего тика.

from_msc

[in] Дата, начиная с которой запрашиваются тики. Указывается в миллисекундах с 01.01.1970. Если параметр *from_msc* не указан, то отдаются тики от самого начала истории. Отдаются тики со временем >= from_msc.

to_msc

[in] Дата, по которую запрашиваются тики. Указывается в миллисекундах с 01.01.1970. Отдаются тики со временем <= to_msc. Если параметр *to_msc* не указан, то отдаются все тики до конца истории.

Возвращаемое значение

Количество скопированных тиков либо -1 в случае ошибки. GetLastError() может возвращать следующие ошибки:

- ERR_HISTORY_TIMEOUT - время ожидание синхронизации тиков вышло, функция отдала всё что было.
- ERR_HISTORY_SMALL_BUFFER - статический буфер слишком маленький, отдано столько, сколько поместилось в массив.

© 2000-2025, MetaQuotes Ltd.

- ERR_NOT_ENOUGH_MEMORY – не хватает памяти для получения истории из указаного диапазона в динамический массив тиков. Не удалось выделить нужный объем памяти под массив тиков.

Примечание

Функция CopyTicksRange() предназначена для запроса тиков из строго указанного диапазона, например, за конкретный день истории. В то время как CopyTicks() позволяет указать только начальную дату, например – получить все тики с начала месяца по текущий момент.

Смотри также

SymbolInfoTick, Структура для получения текущих цен, OnTick, CopyTicks

© 2000-2025, MetaQuotes Ltd.

iBars

Возвращает количество баров в истории по соответствующему символу и периоду.

```
int  iBars(
   const string           symbol,          // символ
   ENUM_TIMEFRAMES        timeframe        // период
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

Возвращаемое значение

Количество баров в истории по соответствующему символу и периоду, но не более чем задано в настройках платформы параметром "Макс. баров в окне" ("Max bars in chart")

Пример:

```
  Print("Bar count on the 'EURUSD,H1' is ",iBars("EURUSD",PERIOD_H1));
```

Смотри также

Bars

© 2000-2025, MetaQuotes Ltd.

iBarShift

Поиск бара по времени. Функция возвращает индекс бара, в который попадает указанное время.

```
int  iBarShift(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   datetime             time,            // время
   bool                 exact=false      // режим
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. PERIOD_CURRENT означает период текущего графика.

time

[in] Значение времени для поиска.

exact=false

[in] Возвращаемое значение, если бар на указанное время не найден. При значении *exact=false iBarShift* возвращает индекс ближайшего бара, у которого время открытия меньше указанного (*time_open<time*). Если такой бар не найден (нет истории раньше указанного времени), то функция вернет -1. Если *exact=true*, то ближайший бар не ищется и функция iBarShift сразу возвращает -1.

Возвращаемое значение

Индекс бара, в который попадает указанное время. Если для указанного времени бар отсутствует ("дыра" в истории), то функция возвращает -1 или индекс ближайшего бара (в зависимости от параметра *exact*).

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- дата приходится на воскресенье
   datetime time=D'2002.04.25 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- если бара на указанное время нет - iBarShift вернет индекс ближайшего бара
   int bar_index=iBarShift(symbol,tf,time,exact);
//--- проверим код ошибки после вызова iBarShift()
   int error=GetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   if(error!=0)
     {
      PrintFormat("iBarShift(): GetLastError=%d - запрашиваемая дата %s "+
                  "для %s %s в доступной истории не найдена",
                  error,TimeToString(time),symbol,EnumToString(tf));
      return;
     }
//--- функция iBarShift() выполнена успешно, выведем результаты для exact=false
   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));
//--- требуем найти индекс бара на указанное время, если его нет, то получим -1
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
//--- функция iBarShift() выполнена успешно, выведем результаты для exact=true
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time)
               ,DayOfWeek(time),bar_index,string(exact));
  }
//+------------------------------------------------------------------+
//| Возвращает название дня недели                                   |
//+------------------------------------------------------------------+
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;
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
/* Результат выполнения
   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)
*/
```

© 2000-2025, MetaQuotes Ltd.

iClose

Возвращает значение цены закрытия бара (указанного параметром shift) соответствующего графика.

```
double  iClose(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   int                  shift            // сдвиг
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

shift

[in] Индекс получаемого значения из таймсерии (сдвиг относительно текущего бара на указанное количество баров назад).

Возвращаемое значение

Значение цены закрытия бара (указанного параметром shift) соответствующего графика или 0 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Примечание

Функция всегда возвращает актуальные данные, для этого она при каждом вызове делает запрос к таймсерии по указанным символу/периоду. Это означает, что при отсутствии готовых данных на первом вызове функции может понадобиться некоторое время для подготовки результата выполнения.

Функция не хранит результатов предыдущих вызовов, локального кеша для быстрого возврата значения нет.

Пример:

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   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"
           );
  }
```

Смотри также

CopyClose, CopyRates

© 2000-2025, MetaQuotes Ltd.

iHigh

Возвращает значение максимальной цены бара (указанного параметром shift) соответствующего графика.

```
double  iHigh(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   int                  shift            // сдвиг
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

shift

[in] Индекс получаемого значения из таймсерии (сдвиг относительно текущего бара на указанное количество баров назад).

Возвращаемое значение

Значение максимальной цены бара (указанного параметром shift) соответствующего графика или 0 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Примечание

Функция всегда возвращает актуальные данные, для этого она при каждом вызове делает запрос к таймсерии по указанным символу/периоду. Это означает, что при отсутствии готовых данных на первом вызове функции может понадобиться некоторое время для подготовки результата выполнения.

Функция не хранит результатов предыдущих вызовов, локального кеша для быстрого возврата значения нет.

Пример:

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   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"
           );
  }
```

Смотри также

CopyHigh, CopyRates

© 2000-2025, MetaQuotes Ltd.

iHighest

Возвращает индекс наибольшего найденного значения (смещение относительно текущего бара) соответствующего графика.

```
int  iHighest(
   const string        symbol,              // символ
   ENUM_TIMEFRAMES     timeframe,           // период
   ENUM_SERIESMODE     type,                // идентификатор таймсерии
   int                 count=WHOLE_ARRAY,   // число элементов
   int                 start=0              // индекс
  );
```

Параметры

symbol

[in] Символ, на котором будет производиться поиск. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

type

[in] Идентификатор таймсерии, в которой будет производится поиск. Может быть любым из значений ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] Число элементов таймсерии (в направлении от текущего бара в сторону возрастания индекса), среди которых должен быть произведен поиск.

start=0

[in] Индекс (смещение относительно текущего бара) начального бара, с которого начинается поиск наибольшего значения. Отрицательные значения игнорируются и заменяются нулевым значением.

Возвращаемое значение

Индекс наибольшего найденного значения (смещение относительно текущего бара) соответствующего графика или -1 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Пример:

```
   double val;
//--- расчет максимального значения цены Close на 20 последовательных барах
//--- начиная с индекса 4 и заканчивая по индекс 23 включительно на текущем графике
   int val_index=iHighest(NULL,0,MODE_CLOSE,20,4);
   if(val_index!=-1)
      val=High[val_index];
   else
      PrintFormat("Ошибка вызова iHighest(). Код ошибки=%d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

iLow

Возвращает значение минимальной цены бара (указанного параметром shift) соответствующего графика.

```
double  iLow(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   int                  shift            // сдвиг
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

shift

[in] Индекс получаемого значения из таймсерии (сдвиг относительно текущего бара на указанное количество баров назад).

Возвращаемое значение

Значение минимальной цены бара (указанного параметром shift) соответствующего графика или 0 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Примечание

Функция всегда возвращает актуальные данные, для этого она при каждом вызове делает запрос к таймсерии по указанным символу/периоду. Это означает, что при отсутствии готовых данных на первом вызове функции может понадобиться некоторое время для подготовки результата выполнения.

Функция не хранит результатов предыдущих вызовов, локального кеша для быстрого возврата значения нет.

Пример:

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   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"
           );
  }
```

Смотри также

CopyLow, CopyRates

© 2000-2025, MetaQuotes Ltd.

iLowest

Возвращает индекс наименьшего найденного значения (смещение относительно текущего бара) соответствующего графика.

```
int  iLowest(
   const string        symbol,              // символ
   ENUM_TIMEFRAMES     timeframe,           // период
   ENUM_SERIESMODE     type,                // идентификатор таймсерии
   int                 count=WHOLE_ARRAY,   // число элементов
   int                 start=0              // индекс
  );
```

Параметры

symbol

[in] Символ, на котором будет производиться поиск. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

type

[in] Идентификатор таймсерии, в которой будет производится поиск. Может быть любым из значений ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] Число элементов таймсерии (в направлении от текущего бара в сторону возрастания индекса), среди которых должен быть произведен поиск.

start=0

[in] Индекс (смещение относительно текущего бара) начального бара, с которого начинается поиск наименьшего значения. Отрицательные значения игнорируются и заменяются нулевым значением.

Возвращаемое значение

Индекс наименьшего найденного значения (смещение относительно текущего бара) соответствующего графика или -1 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Пример:

```
   double val;
//--- поиск бара с минимальным значением реального объема на 15 последовательных барах
//--- начиная с индекса 10 и заканчивая по индекс 24 включительно на текущем графике
   int val_index=iLowest(NULL,0,MODE_REAL_VOLUME,15,10);
   if(val_index!=-1)
      val=Low[val_index];
   else
      PrintFormat("Ошибка вызова iLowest(). Код ошибки=%d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

iOpen

Возвращает значение цены открытия бара (указанного параметром shift) соответствующего графика.

```
double  iOpen(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   int                  shift            // сдвиг
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

shift

[in] Индекс получаемого значения из таймсерии (сдвиг относительно текущего бара на указанное количество баров назад).

Возвращаемое значение

Значение цены открытия бара (указанного параметром shift) соответствующего графика или 0 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Примечание

Функция всегда возвращает актуальные данные, для этого она при каждом вызове делает запрос к таймсерии по указанным символу/периоду. Это означает, что при отсутствии готовых данных на первом вызове функции может понадобиться некоторое время для подготовки результата выполнения.

Функция не хранит результатов предыдущих вызовов, локального кеша для быстрого возврата значения нет.

Пример:

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   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"
           );
  }
```

Смотри также

CopyOpen, CopyRates

© 2000-2025, MetaQuotes Ltd.

iTime

Возвращает значение времени открытия бара (указанного параметром shift) соответствующего графика.

```
datetime  iTime(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   int                  shift            // сдвиг
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

shift

[in] Индекс получаемого значения из таймсерии (сдвиг относительно текущего бара на указанное количество баров назад).

Возвращаемое значение

Значение времени открытия бара (указанного параметром shift) соответствующего графика или 0 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Примечание

Функция всегда возвращает актуальные данные, для этого она при каждом вызове делает запрос к таймсерии по указанным символу/периоду. Это означает, что при отсутствии готовых данных на первом вызове функции может понадобиться некоторое время для подготовки результата выполнения.

Функция не хранит результатов предыдущих вызовов, локального кеша для быстрого возврата значения нет

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- дата приходится на воскресенье
   datetime time=D'2018.06.10 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- бара на указанное время нет - поэтому iBarShift вернет индекс ближайшего бара
```

© 2000-2025, MetaQuotes Ltd.

```
   int bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               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));
//--- требуем найти индекс бара на указанное время, но его нет - вернем -1
   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
  }
//+------------------------------------------------------------------+
//| Возвращает название дня недели                                   |
//+------------------------------------------------------------------+
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;
  }
/* Результат:
   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)
*/
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

CopyTime, CopyRates

© 2000-2025, MetaQuotes Ltd.

iTickVolume

Возвращает значение тикового объема бара (указанного параметром shift) соответствующего графика.

```
long  iTickVolume(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   int                  shift            // сдвиг
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

shift

[in] Индекс получаемого значения из таймсерии (сдвиг относительно текущего бара на указанное количество баров назад).

Возвращаемое значение

Значение тикового объема бара (указанного параметром shift) соответствующего графика или 0 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Примечание

Функция всегда возвращает актуальные данные, для этого она при каждом вызове делает запрос к таймсерии по указанным символу/периоду. Это означает, что при отсутствии готовых данных на первом вызове функции может понадобиться некоторое время для подготовки результата выполнения.

Функция не хранит результатов предыдущих вызовов, локального кеша для быстрого возврата значения нет.

Пример:

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   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"
           );
  }
```

Смотри также

CopyTickVolume, CopyRates

© 2000-2025, MetaQuotes Ltd.

iRealVolume

Возвращает значение реального объема бара (указанного параметром shift) соответствующего графика.

```
long  iRealVolume(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   int                  shift            // сдвиг
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

shift

[in] Индекс получаемого значения из таймсерии (сдвиг относительно текущего бара на указанное количество баров назад).

Возвращаемое значение

Значение реального объема бара (указанного параметром shift) соответствующего графика или 0 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Примечание

Функция всегда возвращает актуальные данные, для этого она при каждом вызове делает запрос к таймсерии по указанным символу/периоду. Это означает, что при отсутствии готовых данных на первом вызове функции может понадобиться некоторое время для подготовки результата выполнения.

Функция не хранит результатов предыдущих вызовов, локального кеша для быстрого возврата значения нет.

Пример:

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   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"
           );
  }
```

Смотри также

CopyRealVolume, CopyRates

© 2000-2025, MetaQuotes Ltd.

iVolume

Возвращает значение тикового объема бара (указанного параметром shift) соответствующего графика.

```
long  iVolume(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   int                  shift            // сдвиг
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

shift

[in] Индекс получаемого значения из таймсерии (сдвиг относительно текущего бара на указанное количество баров назад).

Возвращаемое значение

Значение тикового объема бара (указанного параметром shift) соответствующего графика или 0 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Примечание

Функция всегда возвращает актуальные данные, для этого она при каждом вызове делает запрос к таймсерии по указанным символу/периоду. Это означает, что при отсутствии готовых данных на первом вызове функции может понадобиться некоторое время для подготовки результата выполнения.

Функция не хранит результатов предыдущих вызовов, локального кеша для быстрого возврата значения нет.

Пример:

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   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"
           );
  }
```

Смотри также

CopyTickVolume, CopyRates

© 2000-2025, MetaQuotes Ltd.

iSpread

Возвращает значение спреда бара (указанного параметром shift) соответствующего графика.

```
long  iSpread(
   const string         symbol,          // символ
   ENUM_TIMEFRAMES      timeframe,       // период
   int                  shift            // сдвиг
   );
```

Параметры

symbol

[in] Символьное имя инструмента. NULL означает текущий символ.

timeframe

[in] Период. Может быть одним из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

shift

[in] Индекс получаемого значения из таймсерии (сдвиг относительно текущего бара на указанное количество баров назад).

Возвращаемое значение

Значение спреда для бара (указанного параметром shift) соответствующего графика или 0 в случае ошибки. Для получения дополнительной информации об ошибке необходимо вызвать функцию GetLastError().

Примечание

Функция всегда возвращает актуальные данные, для этого она при каждом вызове делает запрос к таймсерии по указанным символу/периоду. Это означает, что при отсутствии готовых данных на первом вызове функции может понадобиться некоторое время для подготовки результата выполнения.

Функция не хранит результатов предыдущих вызовов, локального кеша для быстрого возврата значения нет.

Пример:

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   int      bars  = iBars(NULL,0);

   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"
           );
  }
```

Смотри также

CopySpread, CopyRates

© 2000-2025, MetaQuotes Ltd.

Пользовательские символы

Функции для создания и редактирования свойств пользовательских символов.

При подключении терминала к конкретному торговому серверу пользователь получает возможность работать с таймсериями тех финансовых инструментов, которые предоставляет данный брокер. Доступные финансовые инструменты показываются списком символов в окне Market Watch, отдельная группа функций позволяет получать информацию о свойствах символа, торговых сессиях и обновлениях стакана заявок.

Представленная в этом разделе группа функций позволяет создавать свои собственные пользовательские символы. Для этого можно использовать существующие символы торгового сервера, текстовые файлы или внешние источники данных.

Функция	Действие
CustomSymbolCreate	Создает пользовательский символ с указанным именем в указанной группе
CustomSymbolDelete	Удаляет пользовательский символ с указанным именем
CustomSymbolSetInteger	Устанавливает для пользовательского символа значение свойства целочисленного типа
CustomSymbolSetDouble	Устанавливает для пользовательского символа значение свойства вещественного типа
CustomSymbolSetString	Устанавливает для пользовательского символа значение свойства строкового типа
CustomSymbolSetMarginRate	Устанавливает для пользовательского символа коэффициенты взимания маржи в зависимости от типа и направления ордера
CustomSymbolSetSessionQuote	Устанавливает время начала и время окончания указанной котировочной сессии для указанных символа и дня недели
CustomSymbolSetSessionTrade	Устанавливает время начала и время окончания указанной торговой сессии для указанных символа и дня недели
CustomRatesDelete	Удаляет все бары в указанном временном интервале из ценовой истории пользовательского инструмента
CustomRatesReplace	Полностью заменяет ценовую историю пользовательского инструмента в указанном временном интервале данными из массива типа MqlRates
CustomRatesUpdate	Добавляет в историю пользовательского инструмента отсутствующие бары и заменяет существующие бары данными из массива типа MqlRates
CustomTicksAdd	Добавляет в ценовую историю пользовательского инструмента данные из массива типа MqlTick. Пользовательский символ должен быть выбран в окне MarketWatch (Обзор рынка)

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
CustomTicksDelete	Удаляет все тики в указанном временном интервале из ценовой истории пользовательского инструмента
CustomTicksReplace	Полностью заменяет ценовую историю пользовательского инструмента в указанном временном интервале данными из массива типа MqlTick
CustomBookAdd	Передает состояние стакана цен по пользовательскому инструменту

© 2000-2025, MetaQuotes Ltd.

CustomSymbolCreate

Создает пользовательский символ с указанным именем в указанной группе.

```
bool  CustomSymbolCreate(
   const string      symbol_name,          // имя пользовательского символа
   const string      symbol_path="",       // название группы, в которой будет создан си
   const string      symbol_origin=NULL    // имя символа, на основе которого будет созд
   );
```

Параметры

symbol_name

[in] Имя пользовательского символа. Не должно содержать групп или подгрупп, в котором символ находится.

symbol_path=""

[in] Имя группы, в которой создается символ.

symbol_origin=NULL

[in] Имя символа, из которого будут скопированы свойства создаваемого пользовательского символа. После создания пользовательского символа можно изменить любое свойство на нужное значение соответствующими функциями.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Все пользовательские символы создаются в специальном разделе Custom. Если имя группы не задано (параметр *symbol_path* в функции CustomSymbolCreate содержит пустую строку или NULL), то пользовательский символ будет создан в корне раздела Custom. Здесь можно провести аналогию с файловой системой, где группы-подгруппы могут рассматриваться как папки-подпапки

Имя символа и название группы задается только латинскими буквами без знаков препинания, пробелов и спецсимволов (допускаются ".", "_", "&" и "#"). Не рекомендуется использовать в названии символы <, >, :, ", /, |, ?, *.

Имя пользовательского символа должно быть уникальным независимо от названия группы, в котором он создается. Если символ с таким именем уже существует, то функция CustomSymbolCreate() вернёт false, а последующий вызов GetLastError() выдаст код ошибки 5300 (ERR_NOT_CUSTOM_SYMBOL) или 5304 (ERR_CUSTOM_SYMBOL_EXIST).

Длина имени символа не должна превышать 31 знака, в противном случае CustomSymbolCreate() вернёт false и будет взведена ошибка 5302 - ERR_CUSTOM_SYMBOL_NAME_LONG.

Параметр *symbol_path* допускается задавать двумя способами:

- только имя группы без имени пользовательского символа, например - "CFD\\Metals". Лучше всего использовать именно этот вариант, чтобы избегать ошибок.
- либо имя <группы> + разделитель групп "\\"+<имя пользовательского символа>, например - "CFD\\Metals\\Platinum". В этом случае имя группы должно оканчиваться точным именем

© 2000-2025, MetaQuotes Ltd.

пользовательского символа. В случае несовпадения пользовательский символ все равно будет создан, но не в той группе, которая задумывалась. Например, если *symbol_path*="CFD\\Metals\\Platinum" и *symbol_name*="platinum" (ошибка в регистре), то будет создан пользовательский символ с именем "platinum" в группе "Custom\CFD\Metals\Platinum". При этом функция SymbolInfoGetString("platinum",SYMBOL_PATH) вернёт значение "Custom\CFD\Metals\Platinum\platinum".

Необходимо иметь в виду, что свойство SYMBOL_PATH возвращает путь вместе с именем символа на конце. Поэтому нельзя просто так копировать его без изменений, если необходимо создать пользовательский символ в точно такой же группе. В этом случае необходимо отрезать имя символа, чтобы не получить результат, описанный выше.

Если в качестве параметра *symbol_origin* задан несуществующий символ, то пользовательский символ будет создан пустым, как если бы параметр *symbol_origin* не был указан. При этом будет взведена ошибка 4301 - ERR_MARKET_UNKNOWN_SYMBOL.

Длина параметра *symbol_path* не должна превышать 127 знаков с учетом "Custom\\", разделителей групп "\\" и имени символа, если оно указано в конце.

Пример:

```
//+------------------------------------------------------------------+
//|                                          CustomSymbolCreate.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // наименование пользовательского симв
#define   CUSTOM_SYMBOL_PATH     "Forex"        // название группы, в которой будет со
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // наименование символа, на основе кот

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- если пользовательский символ создать не удалось - сообщим об этом в журнале
   if(!CustomSymbolCreate(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL_ORIGIN
     {
      Print("CustomSymbolCreate() failed. Error ", GetLastError());
      return;
     }

//--- проверим существование созданного символа и получим группу, в которой он создан
   bool   custom= false;
   bool   exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
```

© 2000-2025, MetaQuotes Ltd.

```
   string path  = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_PATH);

//--- распечатаем в журнале результат создания символа и название группы - заданное и
   PrintFormat("Custom symbol '%s' created\n"+
               "Symbol '%s' is exist: %s\n"+
               "Symbol '%s' is custom: %s\n"+
               "Path specified in the settings: '%s'\n"+
               "Path returned from the 'SYMBOL_PATH' property: '%s'",
               CUSTOM_SYMBOL_NAME,
               CUSTOM_SYMBOL_NAME, (string)exist,
               CUSTOM_SYMBOL_NAME, (string)custom,
               CUSTOM_SYMBOL_PATH,
               path);

//--- подождем две секунды и удалим созданный символ с сообщением о результате в журна
   Sleep(2000);
   ResetLastError();
   bool deleted = CustomSymbolDelete(CUSTOM_SYMBOL_NAME);
   Print(deleted ? StringFormat("Custom symbol '%s' removed", CUSTOM_SYMBOL_NAME) : St
   /*
   результат:
   Custom symbol 'EURUSD.C' created
   Symbol 'EURUSD.C' is exist: true
   Symbol 'EURUSD.C' is custom: true
   Path specified in the settings: 'Forex'
   Path returned from the 'SYMBOL_PATH' property: 'Custom\Forex\EURUSD.C'
   Custom symbol 'EURUSD.C' removed
   */
  }
```

Смотри также

SymbolName, SymbolSelect, CustomSymbolDelete

© 2000-2025, MetaQuotes Ltd.

CustomSymbolDelete

Удаляет пользовательский символ с указанным именем.

```
bool  CustomSymbolDelete(
   const string      symbol_name          // имя пользовательского символа
   );
```

Параметры

symbol

[in] Имя пользовательского символа. Не должно совпадать с именем уже существующего символа.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Пользовательский символ, отображаемый в обзоре рынка (Market Watch) или по которому открыт график, не может быть удалён.

Пример:

```
//+------------------------------------------------------------------+
//|                                          CustomSymbolDelete.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // наименование пользовательского симв
#define   CUSTOM_SYMBOL_PATH     "Forex"        // название группы, в которой будет со
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // наименование символа, на основе кот

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- если пользовательский символ создать не удалось - сообщим об этом в журнале
   if(!CustomSymbolCreate(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL_ORIGIN
     {
      Print("CustomSymbolCreate() failed. Error ", GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- проверим существование созданного символа и распечатаем в журнале результат
   bool custom= false;
   bool exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   PrintFormat("Custom symbol '%s' exists: %s", CUSTOM_SYMBOL_NAME, (string)exist);

//--- подождем две секунды и удалим созданный символ с сообщением о результате в журна
   Sleep(2000);
   ResetLastError();
   bool deleted = CustomSymbolDelete(CUSTOM_SYMBOL_NAME);
   Print(deleted ? StringFormat("Custom symbol '%s' removed", CUSTOM_SYMBOL_NAME) : St

//--- проверим существование созданного символа и распечатаем в журнале результат
   exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   PrintFormat("Custom symbol '%s' exists: %s", CUSTOM_SYMBOL_NAME, (string)exist);
   /*
   результат:
   Custom symbol 'EURUSD.C' exists: true
   Custom symbol 'EURUSD.C' removed
   Custom symbol 'EURUSD.C' exists: false
   */
  }
```

Смотри также

SymbolName, SymbolSelect, CustomSymbolCreate

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetInteger

Устанавливает для пользовательского символа значение свойства целочисленного типа.

```
bool  CustomSymbolSetInteger(
   const string               symbol_name,      // имя символа
   ENUM_SYMBOL_INFO_INTEGER   property_id,      // идентификатор свойства
   long                       property_value    // значение свойства
   );
```

Параметры

symbol_name

[in] Имя пользовательского символа.

property_id

[in] Идентификатор свойства символа. Значение может быть одним из значений перечисления ENUM_SYMBOL_INFO_INTEGER.

property_value

[in] Переменная типа long, содержащая значение свойства.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Минутная и тиковая история пользовательского символа полностью удаляется, если в спецификации символа изменить любое из этих свойств:

- SYMBOL_CHART_MODE - тип цены, который используется для построения баров (Bid или Last)
- SYMBOL_DIGITS - количество знаков после запятой для отражения цены

После удаления истории пользовательского символа терминал попытается создать новую историю с использованием обновленных свойств. То же самое происходит и при ручном изменении свойств пользовательского символа.

Пример:

```
//+------------------------------------------------------------------+
//|                                      CustomSymbolSetInteger.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // наименование пользовательского симв
#define   CUSTOM_SYMBOL_PATH     "Forex"        // название группы, в которой будет со
```

© 2000-2025, MetaQuotes Ltd.

```
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // наименование символа, на основе ко

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим и распечатаем в журнале свойства символа, на основе которого создан поль
//--- (режим заключения сделок, уровень установки Stop ордеров и дистанция заморозки т
   ENUM_SYMBOL_TRADE_EXECUTION origin_exe_mode = (ENUM_SYMBOL_TRADE_EXECUTION)SymbolIn
   int origin_stops_level = (int)SymbolInfoInteger(CUSTOM_SYMBOL_ORIGIN, SYMBOL_TRADE_
   int origin_freeze_level= (int)SymbolInfoInteger(CUSTOM_SYMBOL_ORIGIN, SYMBOL_TRADE_

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Deal execution mode: %s\n  Stops Level: %d\n  Freeze Level: %d",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               StringSubstr(EnumToString(origin_exe_mode), 23), origin_stops_level, or

//--- установим для этих свойств пользовательского символа иные значения
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_EXEMODE, SYMBOL_TRADE
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_STOPS_LEVEL, 10);
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_FREEZE_LEVEL, 3);

//--- если при установке какого-либо из свойств была ошибка - выведем об этом сообщени
   if(!res)
      Print("CustomSymbolSetInteger() failed. Error ", GetLastError());

//--- получим и распечатаем в журнале изменённые свойства пользовательского символа
//--- (режим заключения сделок, уровень установки Stop ордеров и дистанция заморозки т
   ENUM_SYMBOL_TRADE_EXECUTION custom_exe_mode = (ENUM_SYMBOL_TRADE_EXECUTION)SymbolIn
   int custom_stops_level = (int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_ST
   int custom_freeze_level= (int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_FR

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Deal execution mode: %s\n  Stops Level: %d\n  Freeze Level: %d",
               CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN,
               StringSubstr(EnumToString(custom_exe_mode), 23), custom_stops_level, cu

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
```

© 2000-2025, MetaQuotes Ltd.

```
//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Deal execution mode: INSTANT
     Stops Level: 0
     Freeze Level: 0
   Custom symbol 'EURUSD.C' based on 'EURUSD':
     Deal execution mode: MARKET
     Stops Level: 10
     Freeze Level: 3
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
```

Смотри также

SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetDouble

Устанавливает для пользовательского символа значение свойства вещественного типа.

```
bool  CustomSymbolSetDouble(
   const string               symbol_name,      // имя символа
   ENUM_SYMBOL_INFO_DOUBLE    property_id,      // идентификатор свойства
   double                     property_value    // значение свойства
   );
```

Параметры

symbol_name

[in] Имя пользовательского символа.

property_id

[in] Идентификатор свойства символа. Значение может быть одним из значений перечисления ENUM_SYMBOL_INFO_DOUBLE.

property_value

[in] Переменная типа double, содержащая значение свойства.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Минутная и тиковая история пользовательского символа полностью удаляется, если в спецификации символа изменить любое из этих свойств:

- SYMBOL_POINT - значение одного пункта
- SYMBOL_TRADE_TICK_SIZE - значение одного тика, которое задает минимальное допустимое изменение цены
- SYMBOL_TRADE_TICK_VALUE - стоимость изменения цены в один тик для прибыльной позиции

После удаления истории пользовательского символа терминал попытается создать новую историю с использованием обновленных свойств. То же самое происходит и при ручном изменении свойств пользовательского символа.

Пример:

```
//+------------------------------------------------------------------+
//|                                      CustomSymbolSetDouble.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // наименование пользовательского симв
#define   CUSTOM_SYMBOL_PATH     "Forex"        // название группы, в которой будет со
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // наименование символа, на основе кот

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим и распечатаем в журнале свойства символа, на основе которого создан поль
//--- (минимальный объем, максимальный объем, минимальный шаг изменения объема для зад
   double origin_vol_min = SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN, SYMBOL_VOLUME_MIN);
   double origin_vol_max = SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN, SYMBOL_VOLUME_MAX);
   double origin_vol_step= SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN, SYMBOL_VOLUME_STEP);

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Volume Min: %.2f\n  Volume Max: %.2f\n  Volume Step: %.2f",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               origin_vol_min, origin_vol_max, origin_vol_step);

//--- установим для этих свойств пользовательского символа иные значения
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MIN, 0.1);
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MAX, 1000);
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_STEP, 0.1);

//--- если при установке какого-либо из свойств была ошибка - выведем об этом сообщени
   if(!res)
      Print("CustomSymbolSetDouble() failed. Error ", GetLastError());

//--- получим и распечатаем в журнале изменённые свойства пользовательского символа
//--- (минимальный объем, максимальный объем, минимальный шаг изменения объема для зад
   double custom_vol_min = SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MIN);
   double custom_vol_max = SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MAX);
   double custom_vol_step= SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_STEP);

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Volume Min: %.2f\n  Volume Max: %.2f\n  Volume Step: %.2f",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               custom_vol_min, custom_vol_max, custom_vol_step);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Volume Min: 0.01
     Volume Max: 500.00
     Volume Step: 0.01
   Custom symbol 'EURUSD' based on 'EURUSD.C':
     Volume Min: 0.10
     Volume Max: 1000.00
     Volume Step: 0.10
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
```

Смотри также

SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetString

Устанавливает для пользовательского символа значение свойства строкового типа.

```
bool  CustomSymbolSetString(
   const string              symbol_name,      // имя символа
   ENUM_SYMBOL_INFO_STRING   property_id,      // идентификатор свойства
   string                    property_value    // значение свойства
   );
```

Параметры

symbol_name

[in] Имя пользовательского символа.

property_id

[in] Идентификатор свойства символа. Значение может быть одним из значений перечисления ENUM_SYMBOL_INFO_STRING.

property_value

[in] Переменная типа string, содержащая значение свойства.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Минутная и тиковая история пользовательского символа полностью удаляется, если в спецификации символа изменить свойство SYMBOL_FORMULA, которое задает формулу для построения цены пользовательского символа. После удаления истории пользовательского символа терминал попытается создать новую историю по новой формуле. То же самое происходит и при ручном изменении формулы пользовательского символа.

Пример:

```
//+------------------------------------------------------------------+
//|                                        CustomSymbolSetString.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // наименование пользовательского симв
#define   CUSTOM_SYMBOL_PATH     "Forex"        // название группы, в которой будет со
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // наименование символа, на основе кот

//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим и распечатаем в журнале свойства символа, на основе которого создан поль
//--- (минимальный объем, максимальный объем, минимальный шаг изменения объема для зак
   string origin_basis    = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_BASIS);
   string origin_category = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_CATEGORY);
   string origin_formula  = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_FORMULA);

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Basis: %s\n  Category: %s\n  Formula: %s",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               origin_basis, origin_category, origin_formula);

//--- установим для этих свойств пользовательского символа иные значения
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_BASIS, CUSTOM_SYMBOL_ORIGIN)
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_CATEGORY, "FX");
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_FORMULA, ("1.0 / "+CUSTOM_SY

//--- если при установке какого-либо из свойств была ошибка - выведем об этом сообщени
   if(!res)
      Print("CustomSymbolSetString() failed. Error ", GetLastError());

//--- получим и распечатаем в журнале изменённые свойства пользовательского символа
//--- (минимальный объем, максимальный объем, минимальный шаг изменения объема для зак
   string custom_basis    = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_BASIS);
   string custom_category = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_CATEGORY);
   string custom_formula  = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_FORMULA);

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Basis: %s\n  Category: %s\n  Formula: %s",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               custom_basis, custom_category, custom_formula);

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Sleep(16);
       //--- при нажатии Del, удаляем созданный пользовательский символ
       if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
         {
          if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
             PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
          break;
         }
      }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Basis:
     Category:
     Formula:
   Custom symbol 'EURUSD' based on 'EURUSD.C':
     Basis: EURUSD
     Category: FX
     Formula: 1.0 / EURUSD
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
```

Смотри также

SymbolInfoString

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetMarginRate

Устанавливает для пользовательского символа коэффициенты взимания маржи в зависимости от типа и направления ордера.

```
bool  CustomSymbolSetMarginRate(
   const string       symbol_name,              // имя символа
   ENUM_ORDER_TYPE    order_type,               // тип ордера
   double             initial_margin_rate,      // коэффициент взимания начальной марж
   double             maintenance_margin_rate   // коэффициент взимания поддерживающей
   );
```

Параметры

symbol_name

[in] Имя пользовательского символа.

order_type

[in] Тип ордера.

initial_margin_rate

[in] Переменная типа double со значением коэффициента взимания начальной маржи. Начальная маржа - это размер гарантийной суммы под совершение сделки объемом в 1 лот соответствующего направления. Умножая коэффициент на начальную маржу, мы можем получить размер средств, который будет зарезервирован на счете при размещении ордера указанного типа.

maintenance_margin_rate

[in] Переменная типа double со значением коэффициента взимания поддерживающей маржи. Поддерживающая маржа - это размер минимальной суммы для поддержания открытой позиции объемом в 1 лот соответствующего направления. Умножая коэффициент на поддерживающую маржу, мы можем получить размер средств, который будет зарезервирован на счете после срабатывания ордера указанного типа.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetMarginRate.mq5 |
//|                               Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // наименование пользовательского симв
#define   CUSTOM_SYMBOL_PATH     "Forex"        // название группы, в которой будет со
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // наименование символа, на основе кот

#define   INITIAL_MARGIN_RATE       1.5         // коэффициент взимания начальной марж
#define   MAINTENANCE_MARGIN_RATE   1.5         // коэффициент взимания поддерживающей

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим и распечатаем в журнале свойства символа, на основе которого создан поль
//--- (коэффициенты взимания начальной и поддерживающей маржи для Buy и Sell ордеров)
   double initial_margin_rate_buy=0;
   double maintenance_margin_rate_buy=0;
   double initial_margin_rate_sell=0;
   double maintenance_margin_rate_sell=0;

   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_ORIGIN, ORDER_TYPE_BUY, initial_margin_rate_b
      return;
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_ORIGIN, ORDER_TYPE_SELL,initial_margin_rate_s
      return;

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Buy order initial margin rate: %f\n  Buy order maintenance margin ra
               "  Sell order initial margin rate: %f\n  Sell order maintenance margin
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               initial_margin_rate_buy, maintenance_margin_rate_buy, initial_margin_ra

//--- установим для этих свойств пользовательского символа иные значения
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_BUY, INITIAL_MARGIN_
   res &=CustomSymbolSetMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_SELL,INITIAL_MARGIN_

//--- если при установке какого-либо из свойств была ошибка - выведем об этом сообщени
   if(!res)
      Print("CustomSymbolSetMarginRate() failed. Error ", GetLastError());

//--- получим и распечатаем в журнале изменённые свойства пользовательского символа
//--- (коэффициенты взимания начальной и поддерживающей маржи для Buy и Sell ордеров)
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_BUY, initial_margin_rate_buy
      return;
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_SELL,initial_margin_rate_sel
      return;

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Buy order initial margin rate: %f\n  Buy order maintenance margin ra
               "  Sell order initial margin rate: %f\n  Sell order maintenance margin
               CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN,
               initial_margin_rate_buy, maintenance_margin_rate_buy, initial_margin_ra

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Buy order initial margin rate: 1.000000
     Buy order maintenance margin rate: 0.000000
     Sell order initial margin rate: 1.000000
     Sell order maintenance margin rate: 0.000000
   Custom symbol 'EURUSD.C' based on 'EURUSD':
     Buy order initial margin rate: 1.500000
     Buy order maintenance margin rate: 1.500000
     Sell order initial margin rate: 1.500000
     Sell order maintenance margin rate: 1.500000
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
//+------------------------------------------------------------------+
//| Возвращает  коэффициенты взимания маржи                          |
//+------------------------------------------------------------------+
bool GetSymbolMarginRate(const string symbol, const ENUM_ORDER_TYPE order_type, double
  {
   ResetLastError();
   if(!SymbolInfoMarginRate(symbol, order_type, initial_margin_rate, maintenance_margi
     {
      PrintFormat("%s: SymbolInfoMarginRate(%s, %s) failed. Error %d",__FUNCTION__, sy
      return false;
     }
   return true;
  }
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

SymbolInfoMarginRate

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetSessionQuote

Устанавливает время начала и время окончания указанной котировочной сессии для указанных символа и дня недели.

```
bool  CustomSymbolSetSessionQuote(
   const string      symbol_name,          // имя символа
   ENUM_DAY_OF_WEEK  day_of_week,          // день недели
   uint              session_index,        // номер сессии
   datetime          from,                 // время начала сессии
   datetime          to                    // время окончания сессии
   );
```

Параметры

symbol_name

[in] Имя пользовательского символа.

ENUM_DAY_OF_WEEK

[in] День недели, значение из перечисления ENUM_DAY_OF_WEEK.

uint

[in] Порядковый номер сессии, для которой нужно установить время начала и время окончания. Индексация сессий начинается с 0.

from

[in] Время начала сессии в секундах от 00 часов 00 минут, значение даты в переменной будет проигнорировано.

to

[in] Время окончания сессии в секундах от 00 часов 00 минут, значение даты в переменной будет проигнорировано.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Если сессия с указанным *session_index* уже существует, то функция просто отредактирует начало и конец сессии.

Если для сессии переданы нулевые параметры начала и конца, то есть указаны *from=0* и *to=0*, то соответствующая сессия с индексом *session_index* удаляется, а сама нумерация сессий сдвигается вниз.

Добавлять сессии можно только последовательно, то есть сессию с индексом *session_index=1* можно добавить только в том случае, если уже существует сессия с индексом равным 0. При нарушении этого правила новая сессия не создается, а сама функция вернет значение false.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetSessionQuote.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"           // наименование пользовательс
#define   CUSTOM_SYMBOL_PATH     "Forex"                 // название группы, в которой
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()                // наименование символа, на о

#define   SESSION_0_FROM         D'1970.01.01 00:15:00'  // время начала сессии 0
#define   SESSION_0_TO           D'1970.01.01 11:59:00'  // время окончания сессии 0
#define   SESSION_1_FROM         D'1970.01.01 12:15:00'  // время начала сессии 1
#define   SESSION_1_TO           D'1970.01.01 23:59:00'  // время окончания сессии 1

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- распечатаем заголовок с базовым символом и индексом сессии и
//--- в цикле по дням недели с ПН по ПТ распечатаем в журнале время начала и конца как
   for(int session=0; session<2; session++)
     {
      PrintFormat("Quote session %d of '%s' symbol from which the custom '%s' was crea
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_ORIGIN, (ENUM_DAY_OF_WEEK)day_of_we
     }

//--- в цикле по двум сессиям
   bool res=true;
   for(int session=0; session<2; session++)
     {
      datetime from = SESSION_0_FROM;
      datetime to   = SESSION_0_TO;
      if(session>0)
        {
         from = SESSION_1_FROM;
         to   = SESSION_1_TO;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      //--- для пользовательского символа для каждого дня недели установим время котир
      ResetLastError();
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         res &=CustomSymbolSetSessionQuote(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_
     }

//--- если при установке какой-либо из сессий была ошибка - выведем об этом сообщение
   if(!res)
      Print("CustomSymbolSetSessionQuote() failed. Error ", GetLastError());

//--- распечатаем заголовок с пользовательским символом и индексом сессии и
//--- в цикле по дням недели с ПН по ПТ распечатаем в журнале время начала и конца ка
   for(int session=0; session<2; session++)
     {
      PrintFormat("Quote session %d of custom symbol '%s' based on '%s'", session, CUS
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_of_week
     }

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   Quote session 0 of 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     00:15 - 23:55
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   Quote session 1 of 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     Session not set
   - Tuesday    Session not set
   - Wednesday  Session not set
   - Thursday   Session not set
```

© 2000-2025, MetaQuotes Ltd.

```
   - Friday     Session not set
   Quote session 0 of custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     00:15 - 11:59
   - Tuesday    00:15 - 11:59
   - Wednesday  00:15 - 11:59
   - Thursday   00:15 - 11:59
   - Friday     00:15 - 11:59
   Quote session 1 of custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     12:15 - 23:59
   - Tuesday    12:15 - 23:59
   - Wednesday  12:15 - 23:59
   - Thursday   12:15 - 23:59
   - Friday     12:15 - 23:59
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
//+------------------------------------------------------------------+
//| Выводит в журнал время начала и окончания указанной              |
//| котировочной сессии для указанных символа и дня недели           |
//+------------------------------------------------------------------+
void SymbolInfoSessionQuotePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- объявим переменные для записи начала и конца котировочной сессии
   datetime date_from;  // время начала сессии
   datetime date_to;    // время окончания сессии

//--- создаём наименование дня недели из константы перечисления
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- получаем данные котировочной сессии по символу и дню недели
   if(!SymbolInfoSessionQuote(symbol, day_of_week, session_index, date_from, date_to))
     {
      int err=GetLastError();
      string message=(err==4307 ? StringFormat("- %-10s Session not set", week_day) :
                      StringFormat("SymbolInfoSessionQuote(%s, %s, session %d) failed.
      Print(message);
      return;
     }

//--- выводим в журнал данные по указанной котировочной сессии
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

Смотри также

SymbolInfoSessionQuote, Информация об инструменте, TimeToStruct, Структура даты

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetSessionTrade

Устанавливает время начала и время окончания указанной торговой сессии для указанных символа и дня недели.

```
bool  CustomSymbolSetSessionTrade(
   const string      symbol_name,          // имя символа
   ENUM_DAY_OF_WEEK  day_of_week,          // день недели
   uint              session_index,        // номер сессии
   datetime          from,                 // время начала сессии
   datetime          to                    // время окончания сессии
   );
```

Параметры

symbol_name

[in] Имя пользовательского символа.

ENUM_DAY_OF_WEEK

[in] День недели, значение из перечисления ENUM_DAY_OF_WEEK.

uint

[in] Порядковый номер сессии, для которой нужно установить время начала и время окончания. Индексация сессий начинается с 0.

from

[in] Время начала сессии в секундах от 00 часов 00 минут, значение даты в переменной будет проигнорировано.

to

[in] Время окончания сессии в секундах от 00 часов 00 минут, значение даты в переменной будет проигнорировано.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Если сессия с указанным *session_index* уже существует, то функция просто отредактирует начало и конец сессии.

Если для сессии переданы нулевые параметры начала и конца, то есть указаны *from=0* и *to=0*, то соответствующая сессия с индексом *session_index* удаляется, а сама нумерация сессий сдвигается вниз.

Добавлять сессии можно только последовательно, то есть сессию с индексом *session_index=1* можно добавить только в том случае, если уже существует сессия с индексом равным 0. При нарушении этого правила новая сессия не создается, а сама функция вернет значение false.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetSessionTrade.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"           // наименование пользовательс
#define   CUSTOM_SYMBOL_PATH     "Forex"                 // название группы, в которой
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()                // наименование символа, на о

#define   SESSION_0_FROM         D'1970.01.01 00:15:00'  // время начала сессии 0
#define   SESSION_0_TO           D'1970.01.01 11:59:00'  // время окончания сессии 0
#define   SESSION_1_FROM         D'1970.01.01 12:15:00'  // время начала сессии 1
#define   SESSION_1_TO           D'1970.01.01 23:59:00'  // время окончания сессии 1

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- распечатаем заголовок с базовым символом и индексом сессии и
//--- в цикле по дням недели с ПН по ПТ распечатаем в журнале время начала и конца как
   for(int session=0; session<2; session++)
     {
      PrintFormat("Trade session %d of the '%s' symbol from which the custom '%s' was
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_ORIGIN, (ENUM_DAY_OF_WEEK)day_of_we
     }

//--- в цикле по двум сессиям
   bool res=true;
   for(int session=0; session<2; session++)
     {
      datetime from = SESSION_0_FROM;
      datetime to   = SESSION_0_TO;
      if(session>0)
        {
         from = SESSION_1_FROM;
         to   = SESSION_1_TO;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      //--- для пользовательского символа для каждого дня недели установим время котир
      ResetLastError();
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         res &=CustomSymbolSetSessionQuote(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_
     }

//--- если при установке какой-либо из сессий была ошибка - выведем об этом сообщение
   if(!res)
      Print("CustomSymbolSetSessionTrade() failed. Error ", GetLastError());

//--- распечатаем заголовок с пользовательским символом и индексом сессии и
//--- в цикле по дням недели с ПН по ПТ распечатаем в журнале время начала и конца как
   for(int session=0; session<2; session++)
     {
      PrintFormat("Trade session %d of a custom symbol '%s' based on '%s'", session, C
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_of_week
     }

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   Trade session 0 of the 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     00:15 - 23:55
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   Trade session 1 of the 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     Session not set
   - Tuesday    Session not set
   - Wednesday  Session not set
   - Thursday   Session not set
```

© 2000-2025, MetaQuotes Ltd.

```
   - Friday     Session not set
   Trade session 0 of a custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     00:15 - 11:59
   - Tuesday    00:15 - 11:59
   - Wednesday  00:15 - 11:59
   - Thursday   00:15 - 11:59
   - Friday     00:15 - 11:59
   Trade session 1 of a custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     12:15 - 23:59
   - Tuesday    12:15 - 23:59
   - Wednesday  12:15 - 23:59
   - Thursday   12:15 - 23:59
   - Friday     12:15 - 23:59
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
//+------------------------------------------------------------------+
//| Выводит в журнал время начала и окончания указанной              |
//| торговой сессии для указанных символа и дня недели               |
//+------------------------------------------------------------------+
void SymbolInfoSessionTradePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- объявим переменные для записи начала и конца котировочной сессии
   datetime date_from;  // время начала сессии
   datetime date_to;    // время окончания сессии

//--- создаём наименование дня недели из константы перечисления
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- получаем данные котировочной сессии по символу и дню недели
   if(!SymbolInfoSessionTrade(symbol, day_of_week, session_index, date_from, date_to))
     {
      int err=GetLastError();
      string message=(err==4307 ? StringFormat("- %-10s Session not set", week_day) :
                      StringFormat("SymbolInfoSessionTrade(%s, %s, session %d) failed.
      Print(message);
      return;
     }

//--- выводим в журнал данные по указанной торговой сессии
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

Смотри также

SymbolInfoSessionTrade, Информация об инструменте, TimeToStruct, Структура даты

© 2000-2025, MetaQuotes Ltd.

CustomRatesDelete

Удаляет все бары в указанном временном интервале из ценовой истории пользовательского инструмента.

```
int  CustomRatesDelete(
   const string      symbol,        // имя символа
   datetime          from,          // с какой даты
   datetime          to             // по какую дату
   );
```

Параметры

symbol

[in] Имя пользовательского инструмента.

from

[in] Время первого бара в ценовой истории из указанного диапазона, подлежащего удалению.

to

[in] Время последнего бара в ценовой истории из указанного диапазона, подлежащего удалению.

Возвращаемое значение

Количество удаленных баров либо -1 в случае ошибки.

Пример:

```
//+------------------------------------------------------------------+
//|                                              CustomRatesDelete.mq5 |
//|                                 Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // наименование пользовательского с
#define   CUSTOM_SYMBOL_PATH     "Forex"           // название группы, в которой будет
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // наименование символа, на основе

#define   DATARATES_COUNT        4                 // количество выводимых баров в жур

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
```

© 2000-2025, MetaQuotes Ltd.

```
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим количество баров стандартного символа
   int bars=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);

//--- получим в массив MqlRates данные всех баров минутного таймфрейма стандартного си
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_OR
      return;
     }

//--- установим скопированные данные в минутную историю пользовательского символа
   ResetLastError();
   if(CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates)<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- после обновления исторических данных получим количество баров пользовательского
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- получим в массив MqlRates данные всех баров минутного таймфрейма пользовательско
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- распечатаем в журнале последние DATARATES_COUNT баров минутной истории пользоват
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:", DATARATES_COUNT)
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- удалим два предпоследних бара данных в минутной истории пользовательского симво
   datetime time_from= rates[bars-3].time;
   datetime time_to  = rates[bars-2].time;
   ResetLastError();
   int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, time_from, time_to);
   if(deleted<0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("CustomRatesDelete(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- после удаления двух баров исторических данных, заново получим количество баров и
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- опять получим данные всех оставшихся баров минутного таймфрейма пользовательско
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- распечатаем в журнале последние DATARATES_COUNT баров обновлённой минутной истор
   PrintFormat("\nLast %d bars after applying CustomRatesDelete() with %d deleted bars
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ и его данные
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- удаляем данные баров
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully d

         //--- удаляем тиковые данные
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- удаляем символ
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   Last 4 bars of the custom symbol's minute history:
```

© 2000-2025, MetaQuotes Ltd.

```
                    [time]   [open]   [high]    [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 20:53:00 1.07341 1.07347 1.07336 1.07343            38         0
   [1] 2024.06.18 20:54:00 1.07344 1.07354 1.07344 1.07353            21         0
   [2] 2024.06.18 20:55:00 1.07353 1.07362 1.07351 1.07356            32         0
   [3] 2024.06.18 20:56:00 1.07356 1.07358 1.07352 1.07354            24         0

   Last 4 bars after applying CustomRatesDelete() with 2 deleted bars:
                    [time]   [open]   [high]    [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 20:51:00 1.07357 1.07358 1.07347 1.07349            25         0
   [1] 2024.06.18 20:52:00 1.07349 1.07350 1.07336 1.07341            31         0
   [2] 2024.06.18 20:53:00 1.07341 1.07347 1.07336 1.07343            38         0
   [3] 2024.06.18 20:56:00 1.07356 1.07358 1.07352 1.07354            24         0
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
```

Смотри также

CustomRatesReplace, CustomRatesUpdate, CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomRatesReplace

Полностью заменяет ценовую историю пользовательского инструмента в указанном временном интервале данными из массива типа MqlRates.

```
int  CustomRatesReplace(
   const string     symbol,              // имя символа
   datetime         from,                // с какой даты
   datetime         to,                  // по какую дату
   const MqlRates&  rates[],             // массив с данными, которые необходимо примен
   uint             count=WHOLE_ARRAY    // количество элементов, которые будут исполь
   );
```

Параметры

symbol

[in] Имя пользовательского инструмента.

from

[in] Время первого бара в ценовой истории из указанного диапазона, подлежащего обновлению.

to

[in] Время последнего бара в ценовой истории из указанного диапазона, подлежащего обновлению.

rates[]

[in] Массив исторических данных типа MqlRates для таймфрейма M1.

count=WHOLE_ARRAY

[in] Количество элементов из массива *rates[]*, которые будут использованы для замены. Значение WHOLE_ARRAY означает, что для замены необходимо использовать все элементы массива *rates[]*.

Возвращаемое значение

Количество обновленных баров либо -1 в случае ошибки.

Примечание

Если бар из массива *rates[]* выходит за пределы указанного диапазона, то он игнорируется. Если такой бар уже есть в ценовой истории и входит в заданный диапазон, то он заменяется. Все остальные бары в текущей ценовой истории за пределами указанного диапазона остаются неизменным. Данные в массиве *rates[]* должны быть корректными по ценам OHLC, а время открытия баров соответствовать таймфрейму M1.

Пример:

```
//+------------------------------------------------------------------+
//|                                          CustomRatesReplace.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // наименование пользовательского с
#define   CUSTOM_SYMBOL_PATH     "Forex"           // название группы, в которой будет
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // наименование символа, на основе

#define   DATARATES_COUNT        4                 // количество выводимых баров в жур

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим количество баров стандартного символа
   int bars=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);

//--- получим в массив MqlRates данные всех баров минутного таймфрейма стандартного си
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_OR
      return;
     }

//--- установим скопированные данные в минутную историю пользовательского символа
   ResetLastError();
   if(CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates)<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- после обновления исторических данных получим количество баров пользовательского
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- получим в массив MqlRates данные всех баров минутного таймфрейма пользовательско
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- распечатаем в журнале последние DATARATES_COUNT баров минутной истории пользова
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:", DATARATES_COUNT)
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- изменим два предпоследних бара данных в минутной истории пользовательского симво
   datetime time_from= rates[bars-3].time;
   datetime time_to  = rates[bars-2].time;

//--- в массиве rates сделаем все цены двух предпоследних баров равными ценам открытия
   rates[bars-3].high=rates[bars-3].open;
   rates[bars-3].low=rates[bars-3].open;
   rates[bars-3].close=rates[bars-3].open;

   rates[bars-2].high=rates[bars-2].open;
   rates[bars-2].low=rates[bars-2].open;
   rates[bars-2].close=rates[bars-2].open;

//--- заменим существующие бары данными из изменённого массива rates
   ResetLastError();
   int replaced=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(replaced<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- после изменения двух баров исторических данных, заново получим количество баров
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- опять получим данные всех баров минутного таймфрейма пользовательского символа
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- распечатаем в журнале последние DATARATES_COUNT баров обновлённой минутной истор
   PrintFormat("\nLast %d bars after applying CustomRatesUpdate() with %d replaced bar
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ и его данные
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- удаляем данные баров
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully c

         //--- удаляем тиковые данные
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- удаляем символ
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   Last 4 bars of the custom symbol's minute history:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.07.29 13:37:00 1.08394 1.08396 1.08388 1.08390            16        1
   [1] 2024.07.29 13:38:00 1.08389 1.08400 1.08389 1.08398            35        1
   [2] 2024.07.29 13:39:00 1.08398 1.08410 1.08394 1.08410            29        1
   [3] 2024.07.29 13:40:00 1.08409 1.08414 1.08408 1.08414            14        1

   Last 4 bars after applying CustomRatesUpdate() with 250820 replaced bars:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.07.29 13:37:00 1.08394 1.08396 1.08388 1.08390            16        1
   [1] 2024.07.29 13:38:00 1.08389 1.08389 1.08389 1.08389            35        1
   [2] 2024.07.29 13:39:00 1.08398 1.08398 1.08398 1.08398            29        1
   [3] 2024.07.29 13:40:00 1.08409 1.08414 1.08408 1.08414            14        1
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
```

© 2000-2025, MetaQuotes Ltd.

```
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
```

Смотри также

CustomRatesDelete, CustomRatesUpdate, CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomRatesUpdate

Добавляет в историю пользовательского инструмента отсутствующие бары и заменяет существующие данными из массива типа MqlRates.

```
int  CustomRatesUpdate(
   const string     symbol,              // имя пользовательского символа
   const MqlRates&  rates[],             // массив с данными, которые необходимо приме
   uint             count=WHOLE_ARRAY    // количество элементов, которые будут исполь
   );
```

Параметры

symbol

[in] Имя пользовательского инструмента.

rates[]

[in] Массив исторических данных типа MqlRates для таймфрейма M1.

count=WHOLE_ARRAY

[in] Количество элементов из массива *rates[]*, которые будут использованы для обновления. Значение WHOLE_ARRAY означает, что необходимо использовать все элементы массива *rates[]*.

Возвращаемое значение

Количество обновленных баров либо -1 в случае ошибки.

Примечание

Если бар из массива *rates[]* отсутствует в текущей истории пользовательского инструмента, то он добавляется. Если такой бар уже есть, то он заменяется. Все остальные бары в текущей ценовой истории остаются неизменным. Данные в массиве *rates[]* должны быть корректными по ценам OHLC, а время открытия баров соответствовать таймфрейму M1.

Пример:

```
//+------------------------------------------------------------------+
//|                                            CustomRatesUpdate.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"      // наименование пользовательского с
#define   CUSTOM_SYMBOL_PATH     "Forex"            // название группы, в которой будет
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()           // наименование символа, на основе

#define   DATARATES_COUNT        4                  // количество выводимых баров в жур
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим и распечатаем в журнале количество баров стандартного символа
   int bars_origin=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);
   PrintFormat("The symbol '%s' from which the custom '%s' was created has %d bars of

//--- получим и распечатаем в журнале количество баров пользовательского символа
   int bars_custom=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);
   PrintFormat("Custom symbol '%s' created from symbol '%s' has %d bars of minute hist

//--- получим в массив MqlRates данные всех баров минутного таймфрейма стандартного си
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars_origin, rates)!=bars_origin)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_OR
      return;
     }

//--- установим скопированные данные в минутную историю пользовательского символа
   ResetLastError();
   int updated=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(updated<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- получим и распечатаем в журнале количество баров пользовательского символа после
   bars_custom=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);
   PrintFormat("\nAfter CustomRatesUpdate(), the custom symbol '%s' has %d bars of min

//--- получим в массив MqlRates данные всех баров минутного таймфрейма пользовательско
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars_custom, rates)!=bars_custom)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
     }

//--- распечатаем в журнале последние четыре бара минутной истории пользовательского с
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:", DATARATES_COUNT)
   ArrayPrint(rates, digits, NULL, bars_custom-DATARATES_COUNT, DATARATES_COUNT);

//--- поменяем данные в массиве MqlRates на рассчитанные по формуле 1.0 / SymbolName
   for(int i=0; i<bars_custom; i++)
     {
      rates[i].open  =(rates[i].open !=0  ? 1.0 / rates[i].open  : rates[i].open);
      rates[i].high  =(rates[i].high !=0  ? 1.0 / rates[i].high  : rates[i].high);
      rates[i].low   =(rates[i].low  !=0  ? 1.0 / rates[i].low   : rates[i].low);
      rates[i].close =(rates[i].close!=0  ? 1.0 / rates[i].close : rates[i].close);
     }

//--- установим изменённые данные в минутную историю пользовательского символа
   ResetLastError();
   updated=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(updated<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- заново получим в массив MqlRates данные всех баров минутного таймфрейма пользова
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars_custom, rates)!=bars_custom)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- распечатаем в журнале последние четыре бара обновлённой минутной истории пользов
   Print("\nLast %d bars after changing the custom symbol calculation formula:", DATAR
   ArrayPrint(rates, digits, NULL, bars_custom-DATARATES_COUNT, DATARATES_COUNT);

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ и его данные
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- удаляем данные баров
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
```

© 2000-2025, MetaQuotes Ltd.

```
            PrintFormat("%d history bars of the custom symbol '%s' were successfully c

         //--- удаляем тиковые данные
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- удаляем символ
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   The symbol 'EURUSD' from which the custom 'EURUSD.C' was created has 250488 bars of
   Custom symbol 'EURUSD.C' created from symbol 'EURUSD' has 0 bars of minute history

   After CustomRatesUpdate(), the custom symbol 'EURUSD.C' has 250488 bars of minute h
   Last 4 bars of the custom symbol's minute history:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 11:14:00 1.07235 1.07239 1.07232 1.07239            24        0
   [1] 2024.06.18 11:15:00 1.07238 1.07239 1.07232 1.07235            44        0
   [2] 2024.06.18 11:16:00 1.07234 1.07238 1.07227 1.07234            37        0
   [3] 2024.06.18 11:17:00 1.07234 1.07234 1.07217 1.07225            41        0

   Last 4 bars after changing the custom symbol calculation formula:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 11:14:00 0.93253 0.93250 0.93256 0.93250            24        0
   [1] 2024.06.18 11:15:00 0.93251 0.93250 0.93256 0.93253            44        0
   [2] 2024.06.18 11:16:00 0.93254 0.93251 0.93260 0.93254            37        0
   [3] 2024.06.18 11:17:00 0.93254 0.93254 0.93269 0.93262            41        0
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
```

Смотри также

CustomRatesReplace, CustomRatesDelete, CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomTicksAdd

Добавляет в ценовую историю пользовательского инструмента данные из массива типа MqlTick. Пользовательский символ должен быть выбран в окне MarketWatch (Обзор рынка).

```
bool  CustomTicksAdd(
   const string     symbol,              // имя символа
   const MqlTick&   ticks[],             // массив с тиковыми данными, которые необходи
   uint             count=WHOLE_ARRAY    // количество элементов, которые будут исполь
   );
```

Параметры

symbol

[in] Имя пользовательского инструмента.

ticks[]

[in] Массив тиковых данных типа MqlTick, упорядоченных по времени в порядке возрастания, то есть требуется чтобы ticks[k].time_msc <= ticks[n].time_msc, если k<n.

count=WHOLE_ARRAY

[in] Количество элементов из массива *ticks[]*, которые будут использованы для добавления. Значение WHOLE_ARRAY означает, что необходимо использовать все элементы массива *ticks[]*.

Возвращаемое значение

Количество добавленных тиков либо -1 в случае ошибки.

Примечание

Функция CustomTicksAdd работает только для пользовательских символов, открытых в окне MarketWatch (Обзор рынка). Если символ не выбран в MarketWatch, то для вставки тиков необходимо использовать CustomTicksReplace.

Функция CustomTicksAdd позволяет транслировать тики так, как если бы они приходили от сервера брокера. Данные записываются не напрямую в базу тиков, а отправляются в окно "Обзор рынка". И уже из него терминал сохраняет тики в своей базе. При большом объеме данных, передаваемых за один вызов, функция меняет свое поведение для экономии ресурсов. Если передается более 256 тиков, данные делятся на две части. Первая часть (большая) сразу напрямую записывается в базу тиков (как это делает CustomTicksReplace). Вторая часть, состоящая из последних 128 тиков, передается в окно "Обзор рынка" и после этого сохраняется терминалом в базе.

Структура MqlTick имеет два поля со значением времени - time (время тика в секундах) и time_msc (время тика в миллисекундах) - которые ведут отсчет от 01 января 1970 года. Обработка этих полей в добавляемых тиках производится по следующим правилам в указанном порядке:

1. если значение ticks[k].time_msc!=0, то используем его для заполнения поля ticks[k].time, то есть для тика выставляется время ticks[k].time=ticks[k].time_msc/1000 (деление целочисленное)
2. если ticks[k].time_msc==0 и ticks[k].time!=0, то время в миллисекундах получается умножением на 1000, то есть ticks[k].time_msc=ticks[k].time*1000

© 2000-2025, MetaQuotes Ltd.

3. если ticks[k].time_msc==0 и ticks[k].time==0, то в эти поля записывается текущее время торгового сервера с точностью до миллисекунд на момент вызова функции CustomTicksAdd.

Если значение полей ticks[k].bid, ticks[k].ask, ticks[k].last или ticks[k].volume больше нуля, то в поле ticks[k].flags пишется комбинация соответствующих флагов:

- TICK_FLAG_BID - тик изменил цену бид
- TICK_FLAG_ASK - тик изменил цену аск
- TICK_FLAG_LAST - тик изменил цену последней сделки
- TICK_FLAG_VOLUME - тик изменил объем

Если значение какого-то поля меньше или равно нуля, соответствующий ему флаг не записываются в поле ticks[k].flags.

Флаги TICK_FLAG_BUY и TICK_FLAG_SELL в историю пользовательского инструмента не добавляются.

Пример:

```
//+------------------------------------------------------------------+
//|                                                CustomTicksAdd.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // наименование пользовательского с
#define   CUSTOM_SYMBOL_PATH     "Forex"           // название группы, в которой будет
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // наименование символа, на основе

#define   DATATICKS_TO_COPY      UINT_MAX          // количество копируемых тиков
#define   DATATICKS_TO_PRINT     20                // количество выводимых тиков в жур

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим в массив MqlTick данные тиков стандартного символа
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;

//--- распечатаем время первого и последнего полученных тиков стандартного символа
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- распечатаем в журнале DATATICKS_TO_PRINT последних тиков стандартного символа
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':", DATATICKS_TO_PRINT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- добавляем пользовательский символ в окно MarketWatch (обзор рынка)
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- добавим в ценовую историю пользовательского символа данные из массива тиков
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'", ar
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms", add

//--- получим в массив MqlTick данные тиков пользовательского символа
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- распечатаем время первого и последнего полученных тиков пользовательского симво
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- распечатаем в журнале DATATICKS_TO_PRINT последних тиков пользовательского симв
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':", DATATICKS_TO_PRINT,
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ и его данные
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- удаляем данные баров
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully c

         //--- удаляем тиковые данные
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- удаляем символ
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   Requested 4294967295 ticks to download tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351183943 tic
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.20 21:18:12.010

   The last 20 ticks for the standard symbol 'EURUSD':
     351183923th Tick: 2024.06.20 21:17:46.380 Bid=1.07124 (Info tick)
     351183924th Tick: 2024.06.20 21:17:47.779 Ask=1.07125 Bid=1.07125 (Info tick)
     351183925th Tick: 2024.06.20 21:17:48.584 Ask=1.07124 Bid=1.07124 (Info tick)
     351183926th Tick: 2024.06.20 21:17:49.481 Ask=1.07125 (Info tick)
     351183927th Tick: 2024.06.20 21:17:49.985 Ask=1.07122 Bid=1.07122 (Info tick)
     351183928th Tick: 2024.06.20 21:17:50.482 Ask=1.07124 Bid=1.07124 (Info tick)
     351183929th Tick: 2024.06.20 21:17:51.584 Ask=1.07123 Bid=1.07123 (Info tick)
     351183930th Tick: 2024.06.20 21:17:52.786 Ask=1.07124 Bid=1.07124 (Info tick)
     351183931th Tick: 2024.06.20 21:17:53.487 Ask=1.07125 Bid=1.07125 (Info tick)
     351183932th Tick: 2024.06.20 21:17:53.989 Ask=1.07126 Bid=1.07126 (Info tick)
     351183933th Tick: 2024.06.20 21:17:55.789 Ask=1.07125 Bid=1.07125 (Info tick)
     351183934th Tick: 2024.06.20 21:17:58.495 Ask=1.07126 Bid=1.07126 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
     351183935th Tick: 2024.06.20 21:18:00.102 Bid=1.07126 (Info tick)
     351183936th Tick: 2024.06.20 21:18:00.698 Ask=1.07129 Bid=1.07129 (Info tick)
     351183937th Tick: 2024.06.20 21:18:03.699 Bid=1.07129 (Info tick)
     351183938th Tick: 2024.06.20 21:18:04.699 Ask=1.07128 Bid=1.07128 (Info tick)
     351183939th Tick: 2024.06.20 21:18:05.901 Ask=1.07129 Bid=1.07129 (Info tick)
     351183940th Tick: 2024.06.20 21:18:07.606 Ask=1.07128 Bid=1.07128 (Info tick)
     351183941th Tick: 2024.06.20 21:18:11.512 Ask=1.07127 Bid=1.07127 (Info tick)
     351183942th Tick: 2024.06.20 21:18:12.010 Ask=1.07126 Bid=1.07126 (Info tick)
   ...
   Start of adding 351183943 ticks to the history of the custom symbol 'EURUSD.C'
   Added 351183943 ticks to the history of the custom symbol 'EURUSD.C' in 269890 ms
   ...
   Requested 351183943 ticks to download tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351183943 t
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.20 21:18:12.010

   The last 20 ticks for the custom symbol 'EURUSD.C':
     351183923th Tick: 2024.06.20 21:17:46.380 Ask=1.07124 Bid=1.07124 (Info tick)
     351183924th Tick: 2024.06.20 21:17:47.779 Ask=1.07125 Bid=1.07125 (Info tick)
     351183925th Tick: 2024.06.20 21:17:48.584 Ask=1.07124 Bid=1.07124 (Info tick)
     351183926th Tick: 2024.06.20 21:17:49.481 Ask=1.07125 Bid=1.07125 (Info tick)
     351183927th Tick: 2024.06.20 21:17:49.985 Ask=1.07122 Bid=1.07122 (Info tick)
     351183928th Tick: 2024.06.20 21:17:50.482 Ask=1.07124 Bid=1.07124 (Info tick)
     351183929th Tick: 2024.06.20 21:17:51.584 Ask=1.07123 Bid=1.07123 (Info tick)
     351183930th Tick: 2024.06.20 21:17:52.786 Ask=1.07124 Bid=1.07124 (Info tick)
     351183931th Tick: 2024.06.20 21:17:53.487 Ask=1.07125 Bid=1.07125 (Info tick)
     351183932th Tick: 2024.06.20 21:17:53.989 Ask=1.07126 Bid=1.07126 (Info tick)
     351183933th Tick: 2024.06.20 21:17:55.789 Ask=1.07125 Bid=1.07125 (Info tick)
     351183934th Tick: 2024.06.20 21:17:58.495 Ask=1.07126 Bid=1.07126 (Info tick)
     351183935th Tick: 2024.06.20 21:18:00.102 Ask=1.07126 Bid=1.07126 (Info tick)
     351183936th Tick: 2024.06.20 21:18:00.698 Ask=1.07129 Bid=1.07129 (Info tick)
     351183937th Tick: 2024.06.20 21:18:03.699 Ask=1.07129 Bid=1.07129 (Info tick)
     351183938th Tick: 2024.06.20 21:18:04.699 Ask=1.07128 Bid=1.07128 (Info tick)
     351183939th Tick: 2024.06.20 21:18:05.901 Ask=1.07129 Bid=1.07129 (Info tick)
     351183940th Tick: 2024.06.20 21:18:07.606 Ask=1.07128 Bid=1.07128 (Info tick)
     351183941th Tick: 2024.06.20 21:18:11.512 Ask=1.07127 Bid=1.07127 (Info tick)
     351183942th Tick: 2024.06.20 21:18:12.010 Ask=1.07126 Bid=1.07126 (Info tick)
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
//+------------------------------------------------------------------+
//| Получает указанное количество тиков в массив                     |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- сообщим о начале загрузки исторических данных
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count, sy

//--- сделаем 3 попытки получить тики
   int attempts=0;
   while(attempts<3)
     {
      //--- замерим время старта перед получением тиков
      uint start=GetTickCount();

      //--- запросим тиковую историю с момента 1970.01.01 00:00.001 (параметр from=1 м
```

© 2000-2025, MetaQuotes Ltd.

```
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- выведем информацию о количестве тиков и затраченном времени
         PrintFormat("The tick history for the '%s' symbol is received in the amount c

         //--- если тиковая история синхронизирована, то код ошибки равен нулю - возвр
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d ms.
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- считаем попытки
      attempts++;
      //--- пауза в 1 секунду в ожидании завершения синхронизации тиковой базы
      Sleep(1000);
     }
//--- не удалось скопировать тики за 3 попытки
   return(false);
  }
//+------------------------------------------------------------------+
//| возвращает строковое описание тика                               |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|TIME_MINUTES

//--- проверим флаги тика
   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);

//--- проверим сначала тик на торговые флаги (для CustomTicksAdd() их нет)
   if(buy_tick || sell_tick)
     {
      //--- сформируем вывод для торгового тика
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last, tick
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last, tick
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- для инфо тика сформируем вывод немного иначе
```

© 2000-2025, MetaQuotes Ltd.

```
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- вернем описание тика
   return(desc);
  }
```

Смотри также

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomTicksDelete

Удалаляет все тики в указанном временном интервале из ценовой истории пользовательского инструмента.

```
int  CustomTicksDelete(
   const string      symbol,            // имя символа
   long              from_msc,          // с какой даты
   long              to_msc             // по какую дату
   );
```

Параметры

symbol

[in] Имя пользовательского инструмента.

from_msc

[in] Время первого тика в ценовой истории из указанного диапазона, подлежащего удалению. Время в миллисекундах с 01.01.1970.

to_msc

[in] Время последнего тика в ценовой истории из указанного диапазона, подлежащего удалению. Время в миллисекундах с 01.01.1970.

Возвращаемое значение

Количество удаленных тиков либо -1 в случае ошибки.

Пример:

```
//+------------------------------------------------------------------+
//|                                            CustomTicksDelete.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // наименование пользовательского с
#define   CUSTOM_SYMBOL_PATH     "Forex"           // название группы, в которой будет
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // наименование символа, на основе

#define   DATATICKS_TO_COPY      UINT_MAX          // количество копируемых тиков
#define   DATATICKS_TO_DELETE    10                // количество удаляемых тиков
#define   DATATICKS_TO_PRINT     20                // количество выводимых тиков в жур

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

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим в массив MqlTick данные тиков стандартного символа
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- распечатаем время первого и последнего полученных тиков стандартного символа
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- распечатаем в журнале DATATICKS_TO_PRINT последних тиков стандартного символа
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':", DATATICKS_TO_PRINT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- добавляем пользовательский символ в окно MarketWatch (обзор рынка)
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- добавим в ценовую историю пользовательского символа данные из массива тиков
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'", ar
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms", add

//--- получим в массив MqlTick только что добавленные данные тиков пользовательского с
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- распечатаем время первого и последнего полученных тиков пользовательского символ
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- распечатаем в журнале DATATICKS_TO_PRINT последних тиков пользовательского симв
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':", DATATICKS_TO_PRINT,
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- получим из истории время тика в миллисекундах, с которого будем удалять диапазо
   long time_from=array[total-DATATICKS_TO_DELETE-1].time_msc;

//--- удалим DATATICKS_TO_DELETE диапазон последних тиков пользовательского символа в
   Print("...");
   start=GetTickCount();
   PrintFormat("Start deleting %u ticks in the history of the custom symbol '%s'", DAT
   int deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, time_from, array[total-2].time_ms
   PrintFormat("Deleted %u ticks in the history of the custom symbol '%s' in %u ms", d

//--- получим в массив MqlTick только что изменённые данные тиков пользовательского си
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- распечатаем время первого и последнего тиков пользовательского символа с удалён
   total=(int)array.Size();
   PrintFormat("Time of the first tick from the changed history: %s.%03u, Time of the
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- распечатаем в журнале DATATICKS_TO_PRINT последних тиков пользовательского симв
   PrintFormat("\nThe last %d ticks of custom symbol '%s' with modified history:", DAT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Sleep(16);
       //--- при нажатии Del, удаляем созданный пользовательский символ и его данные
       if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
         {
          //--- удаляем данные баров
          int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
          if(deleted>0)
             PrintFormat("%d history bars of the custom symbol '%s' were successfully c

          //--- удаляем тиковые данные
          deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
          if(deleted>0)
             PrintFormat("%d history ticks of the custom symbol '%s' were successfully

          //--- удаляем символ
          if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
             PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
          break;
         }
      }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   Requested 4294967295 ticks to get tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351199027 tic
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 10:10:40.392

   The last 20 ticks for the standard symbol 'EURUSD':
     351199007th Tick: 2024.06.21 10:10:23.045 Bid=1.07032 (Info tick)
     351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
     351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 (Info tick)
     351199010th Tick: 2024.06.21 10:10:25.146 Bid=1.07032 (Info tick)
     351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
     351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
     351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
     351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
     351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
     351199016th Tick: 2024.06.21 10:10:30.258 Bid=1.07036 (Info tick)
     351199017th Tick: 2024.06.21 10:10:30.872 Ask=1.07035 Bid=1.07035 (Info tick)
     351199018th Tick: 2024.06.21 10:10:31.358 Ask=1.07036 (Info tick)
     351199019th Tick: 2024.06.21 10:10:31.859 Ask=1.07037 Bid=1.07037 (Info tick)
     351199020th Tick: 2024.06.21 10:10:32.377 Ask=1.07039 Bid=1.07039 (Info tick)
     351199021th Tick: 2024.06.21 10:10:32.962 Ask=1.0704 Bid=1.0704 (Info tick)
     351199022th Tick: 2024.06.21 10:10:33.961 Ask=1.07039 Bid=1.07039 (Info tick)
     351199023th Tick: 2024.06.21 10:10:34.667 Ask=1.0704 (Info tick)
     351199024th Tick: 2024.06.21 10:10:35.170 Bid=1.0704 (Info tick)
     351199025th Tick: 2024.06.21 10:10:38.266 Ask=1.07041 Bid=1.07041 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
  351199026th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
...
Start of adding 351199027 ticks to the history of the custom symbol 'EURUSD.C'
Added 351199027 ticks to the history of the custom symbol 'EURUSD.C' in 261594 ms
...
Requested 351199027 ticks to get tick history for the symbol 'EURUSD.C'
The tick history for the 'EURUSD.C' symbol is received in the amount of 351199027 t
First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 10:10:40.392

The last 20 ticks for the custom symbol 'EURUSD.C':
  351199007th Tick: 2024.06.21 10:10:23.045 Ask=1.07032 Bid=1.07032 (Info tick)
  351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
  351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 Bid=1.07032 (Info tick)
  351199010th Tick: 2024.06.21 10:10:25.146 Ask=1.07032 Bid=1.07032 (Info tick)
  351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
  351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
  351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
  351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
  351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
  351199016th Tick: 2024.06.21 10:10:30.258 Ask=1.07036 Bid=1.07036 (Info tick)
  351199017th Tick: 2024.06.21 10:10:30.872 Ask=1.07035 Bid=1.07035 (Info tick)
  351199018th Tick: 2024.06.21 10:10:31.358 Ask=1.07036 Bid=1.07036 (Info tick)
  351199019th Tick: 2024.06.21 10:10:31.859 Ask=1.07037 Bid=1.07037 (Info tick)
  351199020th Tick: 2024.06.21 10:10:32.377 Ask=1.07039 Bid=1.07039 (Info tick)
  351199021th Tick: 2024.06.21 10:10:32.962 Ask=1.0704 Bid=1.0704 (Info tick)
  351199022th Tick: 2024.06.21 10:10:33.961 Ask=1.07039 Bid=1.07039 (Info tick)
  351199023th Tick: 2024.06.21 10:10:34.667 Ask=1.0704 Bid=1.0704 (Info tick)
  351199024th Tick: 2024.06.21 10:10:35.170 Ask=1.0704 Bid=1.0704 (Info tick)
  351199025th Tick: 2024.06.21 10:10:38.266 Ask=1.07041 Bid=1.07041 (Info tick)
  351199026th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
...
Start deleting 10 ticks in the history of the custom symbol 'EURUSD.C'
Deleted 10 ticks in the history of the custom symbol 'EURUSD.C' in 188 ms
...
Requested 351199027 ticks to get tick history for the symbol 'EURUSD.C'
The tick history for the 'EURUSD.C' symbol is received in the amount of 351199017 t
Time of the first tick from the changed history: 2011.12.19 00:00:08.000, Time of t

The last 20 ticks of custom symbol 'EURUSD.C' with modified history:
  351198997th Tick: 2024.06.21 10:10:14.935 Ask=1.07036 Bid=1.07036 (Info tick)
  351198998th Tick: 2024.06.21 10:10:15.533 Ask=1.07035 Bid=1.07035 (Info tick)
  351198999th Tick: 2024.06.21 10:10:17.736 Ask=1.07036 Bid=1.07036 (Info tick)
  351199000th Tick: 2024.06.21 10:10:18.540 Ask=1.07037 Bid=1.07037 (Info tick)
  351199001th Tick: 2024.06.21 10:10:19.046 Ask=1.07038 Bid=1.07038 (Info tick)
  351199002th Tick: 2024.06.21 10:10:19.542 Ask=1.07036 Bid=1.07036 (Info tick)
  351199003th Tick: 2024.06.21 10:10:20.041 Ask=1.07035 Bid=1.07035 (Info tick)
  351199004th Tick: 2024.06.21 10:10:21.041 Ask=1.07035 Bid=1.07035 (Info tick)
  351199005th Tick: 2024.06.21 10:10:21.544 Ask=1.07032 Bid=1.07032 (Info tick)
  351199006th Tick: 2024.06.21 10:10:22.344 Ask=1.07032 Bid=1.07032 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
     351199007th Tick: 2024.06.21 10:10:23.045 Ask=1.07032 Bid=1.07032 (Info tick)
     351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
     351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 Bid=1.07032 (Info tick)
     351199010th Tick: 2024.06.21 10:10:25.146 Ask=1.07032 Bid=1.07032 (Info tick)
     351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
     351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
     351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
     351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
     351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
     351199016th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
//+------------------------------------------------------------------+
//| Получает указанное количество тиков в массив                     |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- сообщим о начале загрузки исторических данных
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count, sy

//--- сделаем 3 попытки получить тики
   int attempts=0;
   while(attempts<3)
     {
      //--- замерим время старта перед получением тиков
      uint start=GetTickCount();

      //--- запросим тиковую историю с момента 1970.01.01 00:00.001 (параметр from=1 m
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- выведем информацию о количестве тиков и затраченном времени
         PrintFormat("The tick history for the '%s' symbol is received in the amount o

         //--- если тиковая история синхронизирована, то код ошибки равен нулю - возвр
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d ms.
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- считаем попытки
      attempts++;
      //--- пауза в 1 секунду в ожидании завершения синхронизации тиковой базы
      Sleep(1000);
     }
//--- не удалось скопировать тики за 3 попытки
   return(false);
  }
//+------------------------------------------------------------------+
//| возвращает строковое описание тика                               |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|TIME_MINUTES
```

© 2000-2025, MetaQuotes Ltd.

```
//--- проверим флаги тика
   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);

//--- проверим сначала тик на торговые флаги (для CustomTicksAdd() их нет)
   if(buy_tick || sell_tick)
     {
      //--- сформируем вывод для торгового тика
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last, tick
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last, tick
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- для инфо тика сформируем вывод немного иначе
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- вернем описание тика
   return(desc);
  }
```

Смотри также

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomTicksReplace

Полностью заменяет ценовую историю пользовательского инструмента в указанном временном интервале данными из массива типа MqlTick.

```
int  CustomTicksReplace(
   const string     symbol,            // имя символа
   long             from_msc,          // с какой даты
   long             to_msc,            // по какую дату
   const MqlTick&   ticks[],           // массив с тиковыми данными, которые необходим
   uint             count=WHOLE_ARRAY  // количество элементов, которые будут использо
   );
```

Параметры

symbol

[in] Имя пользовательского инструмента.

from_msc

[in] Время первого тика в ценовой истории из указанного диапазона, подлежащего удалению. Время в миллисекундах с 01.01.1970.

to_msc

[in] Время последнего тика в ценовой истории из указанного диапазона, подлежащего удалению. Время в миллисекундах с 01.01.1970.

ticks[]

[in] Массив тиковых данных типа MqlTick, упорядоченных по времени в порядке возрастания.

count=WHOLE_ARRAY

[in] Количество элементов из массива *ticks[]*, которые будут использованы для замены в указанном интервале времени. Значение WHOLE_ARRAY означает, что необходимо использовать все элементы массива *ticks[]*.

Возвращаемое значение

Количество обновленных тиков либо -1 в случае ошибки.

Примечание

Так как в потоке котировок нередко несколько тиков могут иметь одно и то же время с точностью до миллисекунды (точное время тика хранится в поле *time_msc* структуры MqlTick), то функция CustomTicksReplace не делает автоматической сортировки элементов массива *ticks[]* по времени. Поэтому массив тиков должен быть предварительно упорядочен по возрастанию времени.

Замена тиков производится последовательно день за днём до времени указанного в *to_msc* либо до момента возникновения ошибки. Сначала обрабатывается первый день из указанного диапазона, затем следующий, и так далее. Как только обнаружится несоответствие времени тика порядку возрастания (неубывания), то процесс замены тиков сразу же прекращается на текущем дне. При этом тики за предыдущие дни будут успешно заменены, а текущий день (на момент неправильного тика) и все оставшиеся дни в указанном интервале останутся без изменения.

© 2000-2025, MetaQuotes Ltd.

Если в массиве *ticks[]* отсутствуют тиковые данные за какой-то день (вообще говоря, интервал любой длительности), то после применения тиковых данных из `ticks[]` в истории пользовательского инструмента образуется "дыра", соответствующая пропущенным данным. Другим словами, вызов CustomTicksReplace с вырезанными тиками за конкретный интервал будет равносильно удалению части тиковой истории как будто вызывается CustomTicksDelete с интервалом "дыры".

Если в базе тиков в указанном интервале времени данные отсутствуют, то CustomTicksReplace просто добавит в нее тики из массива ticks[].

Функция CustomTicksReplace работает напрямую с базой данных тиков.

Пример:

```
//+------------------------------------------------------------------+
//|                                          CustomTicksReplace.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // наименование пользовательского с
#define   CUSTOM_SYMBOL_PATH     "Forex"           // название группы, в которой будет
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // наименование символа, на основе

#define   DATATICKS_TO_COPY      UINT_MAX          // количество копируемых тиков
#define   DATATICKS_TO_PRINT     20                // количество выводимых тиков в жур

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем код ошибки при создании пользовательского символа
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- если код ошибки не 0 (успешное создание символа) и не 5304 (символ уже создан) -
   if(create!=0 && create!=5304)
      return;

//--- получим в массив MqlTick данные тиков стандартного символа
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- распечатаем время первого и последнего полученных тиков стандартного символа
   int total=(int)array.Size();
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- распечатаем в журнале DATATICKS_TO_PRINT последних тиков стандартного символа
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':", DATATICKS_TO_PRINT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- добавляем пользовательский символ в окно MarketWatch (обзор рынка)
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- добавим в ценовую историю пользовательского символа данные из массива тиков
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'", ar
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms", add

//--- получим в массив MqlTick только что добавленные данные тиков пользовательского с
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- распечатаем время первого и последнего полученных тиков пользовательского симво
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- распечатаем в журнале DATATICKS_TO_PRINT последних тиков пользовательского симво
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':", DATATICKS_TO_PRINT,
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- теперь поменяем значения Ask и Bid тиков в массиве по формуле Ask(Symbol) = 1.0
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<total; i++)
     {
      array[i].ask = (array[i].ask !=0 ? 1.0 / array[i].ask : array[i].ask);
      array[i].bid = (array[i].bid !=0 ? 1.0 / array[i].bid : array[i].bid);
     }
   Print("\nNow the ticks are changed");

//--- заменим тиковую историю пользовательского символа данными из изменённого массива
   Print("...");
   start=GetTickCount();
   PrintFormat("Start replacing %u changed ticks in the history of the custom symbol '
   int replaced=CustomTicksReplace(CUSTOM_SYMBOL_NAME, array[0].time_msc, array[total-
   PrintFormat("Replaced %u ticks in the history of the custom symbol '%s' in %u ms",

//--- получим в массив MqlTick только что заменённые данные тиков пользовательского си
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- распечатаем время первого и последнего полученных изменённых тиков пользовательс
   total=(int)array.Size();
   PrintFormat("First changed tick time: %s.%03u, Last changed tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- распечатаем в журнале DATATICKS_TO_PRINT последних изменённых тиков пользовател
   PrintFormat("\nThe last %d changed ticks for the custom symbol '%s':", DATATICKS_TO
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Changed tick: %s", i, GetTickDescription(array[i]));
     }

//--- выведем на график в комментарии подсказку о клавишах завершения работы скрипта
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- в бесконечном цикле ожидаем нажатия клавиш Esc или Del для выхода
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- при нажатии Del, удаляем созданный пользовательский символ и его данные
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- удаляем данные баров
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully c

         //--- удаляем тиковые данные
```

© 2000-2025, MetaQuotes Ltd.

```
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- удаляем символ
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- перед выходом очистим график
   Comment("");
   /*
   результат:
   Requested 4294967295 ticks to get tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351195822 tic
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 08:39:03.113

   The last 20 ticks for the standard symbol 'EURUSD':
     351195802th Tick: 2024.06.21 08:38:10.076 Ask=1.07194 (Info tick)
     351195803th Tick: 2024.06.21 08:38:13.162 Ask=1.07195 (Info tick)
     351195804th Tick: 2024.06.21 08:38:13.872 Bid=1.07195 (Info tick)
     351195805th Tick: 2024.06.21 08:38:14.866 Ask=1.07194 Bid=1.07194 (Info tick)
     351195806th Tick: 2024.06.21 08:38:17.374 Bid=1.07194 (Info tick)
     351195807th Tick: 2024.06.21 08:38:18.883 Bid=1.07194 (Info tick)
     351195808th Tick: 2024.06.21 08:38:19.771 Bid=1.07194 (Info tick)
     351195809th Tick: 2024.06.21 08:38:20.873 Ask=1.07195 Bid=1.07195 (Info tick)
     351195810th Tick: 2024.06.21 08:38:22.278 Ask=1.07196 Bid=1.07196 (Info tick)
     351195811th Tick: 2024.06.21 08:38:22.775 Bid=1.07196 (Info tick)
     351195812th Tick: 2024.06.21 08:38:23.477 Bid=1.07196 (Info tick)
     351195813th Tick: 2024.06.21 08:38:38.194 Ask=1.07197 (Info tick)
     351195814th Tick: 2024.06.21 08:38:38.789 Ask=1.07196 (Info tick)
     351195815th Tick: 2024.06.21 08:38:39.290 Ask=1.07197 (Info tick)
     351195816th Tick: 2024.06.21 08:38:43.695 Ask=1.07196 (Info tick)
     351195817th Tick: 2024.06.21 08:38:52.203 Ask=1.07195 Bid=1.07195 (Info tick)
     351195818th Tick: 2024.06.21 08:38:55.105 Ask=1.07196 Bid=1.07196 (Info tick)
     351195819th Tick: 2024.06.21 08:38:57.607 Ask=1.07195 Bid=1.07195 (Info tick)
     351195820th Tick: 2024.06.21 08:39:00.512 Ask=1.07196 Bid=1.07196 (Info tick)
     351195821th Tick: 2024.06.21 08:39:03.113 Ask=1.07195 Bid=1.07195 (Info tick)
   ...
   Start of adding 351195822 ticks to the history of the custom symbol 'EURUSD.C'
   Added 351195822 ticks to the history of the custom symbol 'EURUSD.C' in 349407 ms
   ...
   Requested 351195822 ticks to get tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351195822 t
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 08:39:03.113

   The last 20 ticks for the custom symbol 'EURUSD.C':
     351195802th Tick: 2024.06.21 08:38:10.076 Ask=1.07194 Bid=1.07194 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
   351195803th Tick: 2024.06.21 08:38:13.162 Ask=1.07195 Bid=1.07195 (Info tick)
   351195804th Tick: 2024.06.21 08:38:13.872 Ask=1.07195 Bid=1.07195 (Info tick)
   351195805th Tick: 2024.06.21 08:38:14.866 Ask=1.07194 Bid=1.07194 (Info tick)
   351195806th Tick: 2024.06.21 08:38:17.374 Ask=1.07194 Bid=1.07194 (Info tick)
   351195807th Tick: 2024.06.21 08:38:18.883 Ask=1.07194 Bid=1.07194 (Info tick)
   351195808th Tick: 2024.06.21 08:38:19.771 Ask=1.07194 Bid=1.07194 (Info tick)
   351195809th Tick: 2024.06.21 08:38:20.873 Ask=1.07195 Bid=1.07195 (Info tick)
   351195810th Tick: 2024.06.21 08:38:22.278 Ask=1.07196 Bid=1.07196 (Info tick)
   351195811th Tick: 2024.06.21 08:38:22.775 Ask=1.07196 Bid=1.07196 (Info tick)
   351195812th Tick: 2024.06.21 08:38:23.477 Ask=1.07196 Bid=1.07196 (Info tick)
   351195813th Tick: 2024.06.21 08:38:38.194 Ask=1.07197 Bid=1.07197 (Info tick)
   351195814th Tick: 2024.06.21 08:38:38.789 Ask=1.07196 Bid=1.07196 (Info tick)
   351195815th Tick: 2024.06.21 08:38:39.290 Ask=1.07197 Bid=1.07197 (Info tick)
   351195816th Tick: 2024.06.21 08:38:43.695 Ask=1.07196 Bid=1.07196 (Info tick)
   351195817th Tick: 2024.06.21 08:38:52.203 Ask=1.07195 Bid=1.07195 (Info tick)
   351195818th Tick: 2024.06.21 08:38:55.105 Ask=1.07196 Bid=1.07196 (Info tick)
   351195819th Tick: 2024.06.21 08:38:57.607 Ask=1.07195 Bid=1.07195 (Info tick)
   351195820th Tick: 2024.06.21 08:39:00.512 Ask=1.07196 Bid=1.07196 (Info tick)
   351195821th Tick: 2024.06.21 08:39:03.113 Ask=1.07195 Bid=1.07195 (Info tick)

Now the ticks are changed
...
Start replacing 351195822 changed ticks in the history of the custom symbol 'EURUSD
Replaced 351195822 ticks in the history of the custom symbol 'EURUSD.C' in 452266 m
...
Requested 351195822 ticks to get tick history for the symbol 'EURUSD.C'
The tick history for the 'EURUSD.C' symbol is received in the amount of 351195822 t
First changed tick time: 2011.12.19 00:00:08.000, Last changed tick time: 2024.06.2

The last 20 changed ticks for the custom symbol 'EURUSD.C':
   351195802th Changed tick: 2024.06.21 08:38:10.076 Ask=0.93289 Bid=0.93289 (Info t
   351195803th Changed tick: 2024.06.21 08:38:13.162 Ask=0.93288 Bid=0.93288 (Info t
   351195804th Changed tick: 2024.06.21 08:38:13.872 Ask=0.93288 Bid=0.93288 (Info t
   351195805th Changed tick: 2024.06.21 08:38:14.866 Ask=0.93289 Bid=0.93289 (Info t
   351195806th Changed tick: 2024.06.21 08:38:17.374 Ask=0.93289 Bid=0.93289 (Info t
   351195807th Changed tick: 2024.06.21 08:38:18.883 Ask=0.93289 Bid=0.93289 (Info t
   351195808th Changed tick: 2024.06.21 08:38:19.771 Ask=0.93289 Bid=0.93289 (Info t
   351195809th Changed tick: 2024.06.21 08:38:20.873 Ask=0.93288 Bid=0.93288 (Info t
   351195810th Changed tick: 2024.06.21 08:38:22.278 Ask=0.93287 Bid=0.93287 (Info t
   351195811th Changed tick: 2024.06.21 08:38:22.775 Ask=0.93287 Bid=0.93287 (Info t
   351195812th Changed tick: 2024.06.21 08:38:23.477 Ask=0.93287 Bid=0.93287 (Info t
   351195813th Changed tick: 2024.06.21 08:38:38.194 Ask=0.93286 Bid=0.93286 (Info t
   351195814th Changed tick: 2024.06.21 08:38:38.789 Ask=0.93287 Bid=0.93287 (Info t
   351195815th Changed tick: 2024.06.21 08:38:39.290 Ask=0.93286 Bid=0.93286 (Info t
   351195816th Changed tick: 2024.06.21 08:38:43.695 Ask=0.93287 Bid=0.93287 (Info t
   351195817th Changed tick: 2024.06.21 08:38:52.203 Ask=0.93288 Bid=0.93288 (Info t
   351195818th Changed tick: 2024.06.21 08:38:55.105 Ask=0.93287 Bid=0.93287 (Info t
   351195819th Changed tick: 2024.06.21 08:38:57.607 Ask=0.93288 Bid=0.93288 (Info t
   351195820th Changed tick: 2024.06.21 08:39:00.512 Ask=0.93287 Bid=0.93287 (Info t
```

© 2000-2025, MetaQuotes Ltd.

```
     351195821th Changed tick: 2024.06.21 08:39:03.113 Ask=0.93288 Bid=0.93288 (Info t
   */
  }
//+------------------------------------------------------------------+
//| Создаёт пользовательский символ, возвращает код ошибки           |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- определяем наименование символа, на основе которого будет создан пользовательски
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- если пользовательский символ создать не удалось, и это не ошибка 5304 - сообщаем
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- успешно
   return(error);
  }
//+------------------------------------------------------------------+
//| Удаляет пользовательский символ                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- скроем символ из окна Обзор рынка
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- если пользовательский символ удалить не удалось - сообщаем об этом в журнале и в
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- успешно
   return(true);
  }
//+------------------------------------------------------------------+
//| Получает указанное количество тиков в массив                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- сообщим о начале загрузки исторических данных
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count, sy

//--- сделаем 3 попытки получить тики
   int attempts=0;
   while(attempts<3)
     {
      //--- замерим время старта перед получением тиков
      uint start=GetTickCount();

      //--- запросим тиковую историю с момента 1970.01.01 00:00.001 (параметр from=1 r
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- выведем информацию о количестве тиков и затраченном времени
         PrintFormat("The tick history for the '%s' symbol is received in the amount o

         //--- если тиковая история синхронизирована, то код ошибки равен нулю - возвр
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d ms.
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- считаем попытки
      attempts++;
      //--- пауза в 1 секунду в ожидании завершения синхронизации тиковой базы
      Sleep(1000);
     }
//--- не удалось скопировать тики за 3 попытки
   return(false);
  }
//+------------------------------------------------------------------+
//| возвращает строковое описание тика                               |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|TIME_MINUTES

//--- проверим флаги тика
   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);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- проверим сначала тик на торговые флаги (для CustomTicksAdd() их нет)
   if(buy_tick || sell_tick)
     {
      //--- сформируем вывод для торгового тика
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last, tick
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last, tick
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- для инфо тика сформируем вывод немного иначе
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- вернем описание тика
   return(desc);
  }
```

Смотри также

CustomRatesDelete, CustomRatesUpdate, CustomTicksDelete, CopyTicks, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomBookAdd

Передает состояние стакана цен по пользовательскому инструменту. Функция позволяет транслировать стакан цен так, как если бы он приходил от сервера брокера.

```
bool  CustomBookAdd(
   const string        symbol,            // имя символа
   const MqlBookInfo&  books[]            // массив с описаниями элементов стакана цен
   uint                count=WHOLE_ARRAY  // количество элементов, которые будут доба
   );
```

Параметры

symbol

[in] Имя пользовательского инструмента.

books[]

[in] Массив данных типа MqlBookInfo, полностью описывающих состояние стакана цен — все заявки на покупку и продажу. Переданное состояние стакана цен полностью заменяет предыдущее.

count=WHOLE_ARRAY

[in] Количество элементов массива *books[]*, которое должно быть передано в функцию. По умолчанию используется весь массив.

Возвращаемое значение

true - в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция CustomBookAdd работает только для пользовательских символов, по которым открыт стакан цен — через интерфейс платформы или функцию MarketBookAdd.

При вбросе стакана цен Bid и Ask инструмента не обновляются. Вы должны самостоятельно контролировать изменение лучших цен и вбрасывать тики при помощи CustomTicksAdd.

Передаваемые данные проверяются на корректность: для каждого элемента должны быть указаны тип, цена и объем. При этом MqlBookInfo.volume и MqlBookInfo.volume_real не должны быть нулевыми или отрицательными, если оба объема отрицательны, то это будет считаться ошибкой. Указывать можно любой из объемов или оба — берется тот, который указан или является положительным:

```
volume=-1 && volume_real=2 — будет использован volume_real=2,
volume=3 && volume_real=0 — будет использован volume=3.
```

Объем с повышенной точностью MqlBookInfo.volume_real имеет больший приоритет по сравнению с обычным MqlBookInfo.volume. Если для элемента стакана указаны оба значения, будет использовано volume_real.

Если хотя бы один элемент стакана описан неверно, система отбросит переданное состояние полностью.

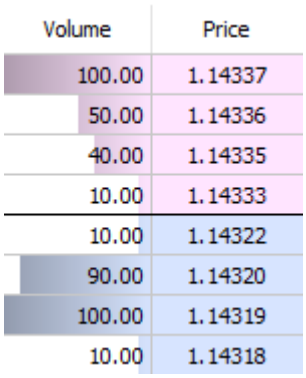
© 2000-2025, MetaQuotes Ltd.

Порядок следования элементов MqlBookInfo в массиве *books* не имеет значения. При сохранении данных терминал сортирует их по цене самостоятельно.

При сохранении данных проверяется параметр "Глубина стакана" (SYMBOL_TICKS_BOOKDEPTH) принимающего пользовательского инструмента. Если количество заявок на продажу в передаваемом стакане цен превышает это значение, лишние уровни отбрасываются. Аналогично для заявок на покупку.

Пример заполнения массива *books*:

Состояние стакана цен	Заполнение books[]

	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;

Пример:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- включаем стакан цен для инструмента, из которого будем брать данные
   MarketBookAdd(Symbol());
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| Tick function                                                    |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   MqlTick ticks[];
   ArrayResize(ticks,1);
//--- скопируем текущие цены из обычного инструмента в пользовательский
   if(SymbolInfoTick(Symbol(),ticks[0]))
     {
      string symbol_name=Symbol()+".SYN";
      CustomTicksAdd(symbol_name,ticks);
     }
  }
//+------------------------------------------------------------------+
//| Book function                                                    |
//+------------------------------------------------------------------+
void OnBookEvent(const string &book_symbol)
  {
//--- скопируем текущее состояние стакана цен из обычного инструмента в пользовательск
   if(book_symbol==Symbol())
     {
      MqlBookInfo book_array[];
      if(MarketBookGet(Symbol(),book_array))
        {
         string symbol_name=Symbol()+".SYN";
         CustomBookAdd(symbol_name,book_array);
        }
     }
  }
```

Смотри также

MarketBookAdd, CustomTicksAdd, OnBookEvent

© 2000-2025, MetaQuotes Ltd.

Операции с графиками

Функции для установки свойств графика (ChartSetInteger, ChartSetDouble, ChartSetString) являются асинхронными и служат для отправки графику команд на изменение. При успешном выполнении этих функций команда попадает в общую очередь событий графика. Изменение свойств графика производится в процессе обработки очереди событий данного графика.

По этой причине не следует ожидать немедленного обновления графика после вызова асинхронных функций. Для принудительного обновления внешнего вида и свойств графика используйте функцию ChartRedraw().

Функция	Действие
ChartApplyTemplate	Применяет к указанному графику шаблон из указанного файла
ChartSaveTemplate	Сохраняет текущие настройки графика в шаблон с указанным именем
ChartWindowFind	Возвращает номер подокна, в котором находится индикатор
ChartTimePriceToXY	Преобразует координаты графика из представления время/цена в координаты по оси X и Y
ChartXYToTimePrice	Преобразует координаты X и Y графика в значения время и цена
ChartOpen	Открывает новый график с указанным символом и периодом
ChartClose	Закрывает указанный график
ChartFirst	Возвращает идентификатор первого графика клиентского терминала
ChartNext	Возвращает идентификатор графика, следующего за указанным
ChartSymbol	Возвращает имя символа указанного графика
ChartPeriod	Возвращает значение периода указанного графика
ChartRedraw	Вызывает принудительную перерисовку указанного графика
ChartSetDouble	Задает значение типа double соответствующего свойства указанного графика
ChartSetInteger	Задает значение целочисленного типа (datetime, int, color, bool или char) соответствующего свойства указанного графика
ChartSetString	Задает значение типа string соответствующего свойства указанного графика
ChartGetDouble	Возвращает значение соответствующего свойства указанного графика
ChartGetInteger	Возвращает целочисленное значение соответствующего свойства указанного графика
ChartGetString	Возвращает строковое значение соответствующего свойства указанного графика

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
ChartNavigate	Осуществляет сдвиг указанного графика на указанное количество баров относительно указанной позиции графика
ChartID	Возвращает идентификатор текущего графика
ChartIndicatorAdd	Добавляет на указанное окно графика индикатор с указанным хэндлом.
ChartIndicatorDelete	Удаляет с указанного окна графика индикатор с указанным именем
ChartIndicatorGet	Возвращает хэндл индикатора с указанным коротким именем на указанном окне графика
ChartIndicatorName	Возвращает короткое имя индикатора по номеру в списке индикаторов на указанном окне графика.
ChartIndicatorsTotal	Возвращает количество всех индикаторов, присоединенных к указанному окну графика.
ChartWindowOnDropped	Возвращает номер подокна графика, на которое брошен мышкой данный эксперт, скрипт, объект или индикатор
ChartPriceOnDropped	Возвращает ценовую координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт
ChartTimeOnDropped	Возвращает временную координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт
ChartXOnDropped	Возвращает координату по оси X, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт
ChartYOnDropped	Возвращает координату по оси Y, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт
ChartSetSymbolPeriod	Меняет значения символа и периода указанного графика
ChartScreenShot	Делает снимок указанного графика в формате GIF, PNG или BMP в зависимости от указанного расширения

© 2000-2025, MetaQuotes Ltd.

ChartApplyTemplate

Применяет к графику указанный шаблон. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartApplyTemplate(
   long          chart_id,      // идентификатор графика
   const string  filename       // имя файла с шаблоном
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

filename

[in] Имя файла, содержащего шаблон.

Возвращаемое значение

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Если посредством этой функции из эксперта будет загружен новый шаблон на график, к которому он присоединен, то эксперт будет выгружен и не сможет продолжить работу.

В целях безопасности права на торговлю при применении шаблона к графику могут ограничиваться:

> **Права на торговлю не могут быть повышены при запуске советника путем применения шаблона с помощью функции ChartApplyTemplate().**

Если у mql5-программы, которая вызывает функцию ChartApplyTemplate(), отсутствуют права на торговлю, то эксперт, загруженный при помощи шаблона, также не будет иметь прав на торговлю вне зависимости от настроек шаблона.

Если у mql5-программы, которая вызывает функцию ChartApplyTemplate(), есть права на торговлю, а в настройках шаблона права отсутствуют, то советник, загруженный при помощи шаблона, не будет иметь прав на торговлю.

Использование шаблонов

Средствами языка MQL5 можно задавать множество свойств графика, в том числе устанавливать цвета с помощью функции ChartSetInteger() :

- Цвет фона графика;
- Цвет осей, шкалы и строки OHLC;
- Цвет сетки;
- Цвет объемов и уровней открытия позиций;
- Цвет бара вверх, тени и окантовки тела бычьей свечи;

© 2000-2025, MetaQuotes Ltd.

- Цвет бара вниз, тени и окантовки тела медвежьей свечи;
- Цвет линии графика и японских свечей "Доджи";
- Цвет тела бычьей свечи;
- Цвет тела медвежьей свечи;
- Цвет линии Bid-цены;
- Цвет линии Ask-цены;
- Цвет линии цены последней совершенной сделки (Last);
- Цвет уровней стоп-ордеров (Stop Loss и Take Profit).

Кроме того, на графике может быть множество графических объектов и индикаторов. Достаточно один раз настроить внешний вид графика со всеми необходимыми индикаторами и сохранить его в виде шаблона, чтобы в последствии можно было применять этот шаблон к любому графику.

Функция ChartApplyTemplate() предназначена для использования ранее сохраненного шаблона и может использоваться в любой mql5-программе. Вторым параметром функции ChartApplyTemplate() передается путь к имени файла, в котором находится шаблон. Поиск файла шаблона осуществляется по следующим правилам:

- если в начале пути стоит разделитель обратная косая черта "\" (пишется "\\"), то шаблон ищется относительно пути каталог_данных_терминала\MQL5,
- если обратной косой черты нет, то шаблон ищется относительно исполняемого EX5-файла, в котором происходит вызов функции ChartApplyTemplate();
- если шаблон не найден в первых двух вариантах, то поиск ведется в папке каталог_терминала\Profiles\Templates\.

Здесь каталог_терминала означает папку, из которой запущен клиентский терминала MetaTrader 5, а каталог_данных_терминала означает папку, в которой хранятся изменяемые файлы и ее расположение может зависеть от типа операционной системы, имени пользователя и настроек безопасности компьютера. В общем случае это разные папки, хотя в некоторых случаях могут и совпадать.

Расположение папок каталог_данных_терминала и каталог_терминала можно узнать с помощью функции TerminalInfoString().

```
//--- каталог из которой запущен терминал
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
   Print("Каталог терминала:",terminal_path);
//--- каталог данных терминала, в котором находится папка MQL5 с советниками и индикат
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   Print("Каталог данных терминала:",terminal_data_path);
```

Примеры записи:

```
//--- шаблон ищем в папке каталог_данных_терминала\MQL5\
ChartApplyTemplate(0,"\\first_template.tpl"))
//--- шаблон ищем в папке каталог_исполняемого_EX5_файла\, затем в папке каталог_данны
ChartApplyTemplate(0,"second_template.tpl"))
//--- шаблон ищем в папке каталог_исполняемого_EX5_файла\My_templates\, затем в папке
ChartApplyTemplate(0,"My_templates\\third_template.tpl"))
```

Шаблоны не относятся к ресурсам, их нельзя включать в исполняемый файл EX5.

© 2000-2025, MetaQuotes Ltd.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- пример применения шаблона, расположенного в каталоге \MQL5\Files
   if(FileIsExist("my_template.tpl"))
     {
      Print("Шаблон my_template.tpl найден в каталоге \Files'");
      //--- применим найденный шаблон
      if(ChartApplyTemplate(0,"\\Files\\my_template.tpl"))
        {
         Print("Применили успешно шаблон 'my_template.tpl'");
         //--- принудительно перерисуем график для быстрого показа изменений
         ChartRedraw();
        }
      else
         Print("Не удалось применить шаблон 'my_template.tpl', ошибка ",GetLastError()
     }
   else
     {
      Print("Файл 'my_template.tpl' не найден в папке "
            +TerminalInfoString(TERMINAL_PATH)+"\\MQL5\\Files");
     }
  }
```

Смотри также

Ресурсы

© 2000-2025, MetaQuotes Ltd.

ChartSaveTemplate

Сохраняет текущие настройки графика в шаблон с указанным именем.

```
bool  ChartSaveTemplate(
   long          chart_id,      // идентификатор графика
   const string  filename       // имя файла для сохранения шаблона
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

filename

[in] Имя файла для сохранения шаблона. Расширение ".tpl" будет добавлено к имени файла автоматически, указывать его не требуется. Шаблон сохраняется в папку **каталог_данных**\Profiles\Templates\ и может быть использован также и для ручного применения в терминале. Если шаблон с данным именем уже существует, то его содержимое будет переписано заново.

Возвращаемое значение

При успешном выполнении функция возвращает true, в противном случае возвращает false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Шаблон позволяет сохранить настройки графика со всеми наложенными на него индикаторами и графическими объектами, чтобы затем применить его на другом графике.

Пример:

```
//+------------------------------------------------------------------+
//|                                      Test_ChartSaveTemplate.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 script_show_inputs
//--- input parameters
input string               symbol="GBPUSD";  // символ нового графика
input ENUM_TIMEFRAMES      period=PERIOD_H3; // таймфрейм нового графика
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- сначала набросим на график индикаторы
   int handle;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- подготовим индикатор к использованию
   if(!PrepareZigzag(NULL,0,handle)) return; // попытка не удалась, выходим
//--- набросим индикатор на текущий график, но в отдельное окно
   if(!ChartIndicatorAdd(0,1,handle))
     {
      PrintFormat("Не удалось добавить на график %s/%s индикатор с хэндлом=%d. Код ош
                  _Symbol,
                  EnumToString(_Period),
                  handle,
                  GetLastError());
      //--- досрочно прекращаем работу программы
      return;
     }
//--- прикажем графику обновиться, чтобы сразу увидеть наброшенный индикатор
   ChartRedraw();
//--- найдем два последних перелома зигзага
   double two_values[];
   datetime two_times[];
   if(!GetLastTwoFractures(two_values,two_times,handle))
     {
      PrintFormat("Не удалось найти два последних перелома в индикторе Zigzag!");
      //--- досрочно прекращаем работу программы
      return;
     }
//--- теперь набросим канал стандартного отклонения
   string channel="StdDeviation Channel";
   if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
     {
      PrintFormat("Не удалось создать объект %s. Код ошибки %d",
                  EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
      return;
     }
   else
     {
      //--- канал создан, доопределим вторую опорную точку
      ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
      //--- зададим каналу текст всплывающей подсказки
      ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo from MQL5 Help");
      //--- обновим график
      ChartRedraw();
     }
//--- сохраним полученный результат в шаблон
   ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- откроем новый график и набросим на него сохраненный шаблон
   long new_chart=ChartOpen(symbol,period);
   //--- включим показ всплывающих подсказок для графических объектов
   ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
   if(new_chart!=0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- набросим на новый график сохраненный шаблон
      ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
     }
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Создает хэндл зигзага и обеспечивает готовность его данных       |
//+------------------------------------------------------------------+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
  {
   ResetLastError();
//--- индикатор Zigzag должен находиться в папке каталог_данных_терминала\MQL5\Example
   h=iCustom(sym,tf,"Examples\\Zigzag");
   if(h==INVALID_HANDLE)
     {
      PrintFormat("%s: Не удалось создать хэндл индикатора Zigzag. Код ошибки %d",
                  __FUNCTION__,GetLastError());
      return false;
     }
//--- при создании хэндла индикатора ему требуется время на расчет значений
   int k=0; // количество попыток чтобы дождаться расчета индикатора
//--- ждем расчета в цикле, делаем паузу в 50 миллисекунд, если расчет еще не готов
   while(BarsCalculated(h)<=0)
     {
      k++;
      //--- выведем количество попыток
      PrintFormat("%s: k=%d",__FUNCTION__,k);
      //--- подождем 50 миллисекунд, пока индикатор рассчитается
      Sleep(50);
      //--- если сделано больше 100 попыток, то что-то не так
      if(k>100)
        {
         //--- сообщим о проблеме
         PrintFormat("Не удалось рассчитать индикатор за %d попыток!");
         //--- досрочно прекращаем работу программы
         return false;
        }
     }
//--- все готово, индикатор создан и значения рассчитаны
   return true;
  }
//+------------------------------------------------------------------+
//| Ищет два последних перелома зигзага и помещает в массивы         |
//+------------------------------------------------------------------+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
  {
   double values[];          // массив для получения значений зигзага
   datetime times[];         // массив для получения времени
   int size=100;             // размер массивов
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- копируем 100 последних значений индикатора
   int copied=CopyBuffer(handle,0,0,size,values);
//--- проверим сколько скопировалось
   if(copied<100)
     {
      PrintFormat("%s: Не удалось скопировать %d значений индикатора с хэндлом=%d. Код
                  __FUNCTION__,size,handle,GetLastError());
      return false;
     }
//--- определим порядок доступа к массиву как в таймсерии
   ArraySetAsSeries(values,true);
//--- сюда запишем номера баров, на которых найдены переломы
   int positions[];
//--- зададим размеры массивов
   ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- счетчики
   int i=0,k=0;
//--- начинаем поиск переломов
   while(i<100)
     {
      double v=values[i];
      //--- пустые значения нас не интересуют
      if(v!=0.0)
        {
         //--- запомним номер бара
         positions[k]=i;
         //--- запомним значение зигзага на переломе
         get_values[k]=values[i];
         PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
         //--- увеличим счетчик
         k++;
         //--- если нашли два перелома, то ломаем цикл
         if(k>2) break;
        }
      i++;
     }
//--- определим порядок доступа к массивам как в таймсерии
   ArraySetAsSeries(times,true);   ArraySetAsSeries(get_times,true);
   if(CopyTime(_Symbol,_Period,0,size,times)<=0)
     {
      PrintFormat("%s: Не удалось скопировать %d значений из CopyTime(). Код ошибки %d
                  __FUNCTION__,size,GetLastError());
      return false;
     }
//--- найдем время открытия баров, на которых были 2 последних перелома
   get_times[0]=times[positions[1]];// предпоследнее значение будет записано первым пе
   get_times[1]=times[positions[2]];// третье с конца значение будет вторым переломом
   PrintFormat("%s: первый=%s,  второй=%s",__FUNCTION__,TimeToString(get_times[1]),Tim
```

© 2000-2025, MetaQuotes Ltd.

```
//--- все получилось
   return true;
  }
```

Смотри также

ChartApplyTemplate(), Ресурсы

© 2000-2025, MetaQuotes Ltd.

ChartWindowFind

Возвращает номер подокна, в котором находится индикатор. Существует 2 варианта функции.

1. Функция ищет на указанном графике подокно с указанным "коротким именем" индикатора (короткое имя выводится слева вверху подокна) и в случае удачи возвращает номер подокна.

```
int  ChartWindowFind(
   long     chart_id,                  // идентификатор графика
   string   indicator_shortname        // короткое имя индикатора, см INDICATOR_SHORTN
```

2. Функция должна вызываться из пользовательского индикатора и возвращает номер подокна, в котором этот индикатор работает.

```
int  ChartWindowFind();
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

indicator_shortname

[in] Короткое имя индикатора.

Возвращаемое значение

Номер подокна в случае удачи. Ноль означает главное окно графика. В случае неудачи возвращает -1.

Примечание

Если вызывается второй вариант функции (без параметров) из скрипта или эксперта, то она возвращает -1.

Не следует путать короткое имя индикатора и имя файла, которое указывается при создании индикатора функциями iCustom() и IndicatorCreate(). Если короткое наименование индикатора не задается явным образом, то при компиляции в нем указывается имя файла с исходным кодом индикатора.

Необходимо правильно формировать короткое имя индикатора, которое с помощью функции IndicatorSetString() записывается в свойство INDICATOR_SHORTNAME. Мы рекомендуем, чтобы короткое имя содержало значения входных параметров индикатора, так как идентификация удаляемого с графика индикатора в функции ChartIndicatorDelete() производится именно по короткому имени.

Пример:

```
#property script_show_inputs
//--- input parameters
input string   shortName="MACD(12,26,9)";
//+------------------------------------------------------------------+
//| Сообщает номер окна графика с указанным индикатором              |
//+------------------------------------------------------------------+
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   int window=-1;
//---
   if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
     {
      //--- функция вызвана из индикатора,имя не требуется
      window=ChartWindowFind();
     }
   else
     {
      //--- функция вызвана из эксперта или скрипта
      window=ChartWindowFind(0,short_name);
      if(window==-1) Print(__FUNCTION__+"(): Error = ",GetLastError());
     }
//---
   return(window);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int window=GetIndicatorSubWindowNumber(0,shortName);
   if(window!=-1)
      Print("Индикатор "+shortName+" находится в окне №"+(string)window);
   else
      Print("Индикатор "+shortName+" не найден. window = "+(string)window);
  }
```

Смотри также

ObjectCreate(), ObjectFind()

© 2000-2025, MetaQuotes Ltd.

ChartTimePriceToXY

Преобразует координаты графика из представления время/цена в координаты по оси X и Y.

```
bool  ChartTimePriceToXY(
   long           chart_id,      // идентификатор графика
   int            sub_window,    // номер подокна
   datetime       time,          // время на графике
   double         price,         // цена на графике
   int&           x,             // координата X для времени на графике
   int&           y              // координата Y для цены на графике
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

sub_window

[in] Номер подокна графика. 0 означает главное окно графика.

time

[in] Значение времени на графике, для которого будет получено значение в пикселах на оси X. Начало координат находится в левом верхнем углу главного окна графика.

price

[in] Значение цены на графике, для которого будет получено значение в пикселах на оси Y. Начало координат находится в левом верхнем углу главного окна графика.

x

[out] Переменная, в которую будет получено преобразование времени в координату X.

y

[out] Переменная, в которую будет получено преобразование цены в координату Y.

Возвращаемое значение

Возвращает true в случае успешного выполнения, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
#define   BAR_NUMBER    0  // номер бара, с которого получаем цену и время

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- копируем данные одного бара по индексу BAR_NUMBER
   MqlRates rates[]={};
   if(CopyRates(_Symbol, _Period, BAR_NUMBER, 1, rates)!=1)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("CopyRates() failed for bar %d. Error %d", BAR_NUMBER, GetLastError
       return;
      }

//--- преобразуем полученные цену и время в пиксельные координаты графика
   int x=0, y=0;
   ResetLastError();
   if(!ChartTimePriceToXY(ChartID(), 0, rates[0].time, rates[0].close, x, y))
     {
      Print("ChartTimePriceToXY() failed. Error ", GetLastError());
      return;
     }

//--- распечатаем полученный результат в журнале
   PrintFormat("For bar[%d] with opening time %s and price %.*f, the chart coordinates

   /*
   результат:
   For bar[0] with opening time 2024.08.09 15:06 and price 1.27378, the chart coordina
   */
  }
```

Смотри также

ChartXYToTimePrice()

© 2000-2025, MetaQuotes Ltd.

ChartXYToTimePrice

Преобразует координаты X и Y графика в значения время и цена.

```
bool  ChartXYToTimePrice(
   long           chart_id,      // идентификатор графика
   int            x,             // координата X на графике
   int            y,             // координата Y на графике
   int&           sub_window,    // номер подокна
   datetime&      time,          // время на графике
   double&        price          // цена на графике
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

x

[in] Координата X.

y

[in] Координата Y.

sub_window

[out] Переменная, в которую будет записан номер подокна графика. 0 означает главное окно графика.

time

[out] Значение времени на графике, для которого будет получено значение в пикселах на оси X. Начало координат находится в левом верхнем углу главного окна графика.

price

[out] Значение цены на графике, для которого будет получено значение в пикселах на оси Y. Начало координат находится в левом верхнем углу главного окна графика.

Возвращаемое значение

Возвращает true в случае успешного выполнения, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- выведем параметры события на график
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment(__FUNCTION__,": id=",id," lparam=",lparam," dparam=",dparam," sparam=",spar
//--- если это события клика мышки на графике
   if(id==CHARTEVENT_CLICK)
     {
      //--- подготовим переменные
      int      x     =(int)lparam;
      int      y     =(int)dparam;
      datetime dt    =0;
      double   price =0;
      int      window=0;
      //--- преобразуем координаты X и Y  в терминах дата/время
      if(ChartXYToTimePrice(0,x,y,window,dt,price))
        {
         PrintFormat("Window=%d X=%d  Y=%d  =>  Time=%s  Price=%G",window,x,y,TimeToSt
         //--- сделаем обратное преобразование: (X,Y) => (Time,Price)
         if(ChartTimePriceToXY(0,window,dt,price,x,y))
            PrintFormat("Time=%s  Price=%G  =>  X=%d  Y=%d",TimeToString(dt),price,x,y
         else
            Print("ChartTimePriceToXY return error code: ",GetLastError());
         //--- удаляем линии
         ObjectDelete(0,"V Line");
         ObjectDelete(0,"H Line");
         //--- создаем горизонтальную и вертикальную линии перекрестия
         ObjectCreate(0,"H Line",OBJ_HLINE,window,dt,price);
         ObjectCreate(0,"V Line",OBJ_VLINE,window,dt,price);
         ChartRedraw(0);
        }
      else
         Print("ChartXYToTimePrice return error code: ",GetLastError());
      Print("+------------------------------------------------------------------+");
     }
  }
```

Смотри также

ChartTimePriceToXY()

© 2000-2025, MetaQuotes Ltd.

ChartOpen

Открывает новый график с указанным символом и периодом.

```
long  ChartOpen(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   );
```

Параметры

symbol

[in] Символ графика. NULL означает символ текущего графика (к которому прикреплен эксперт).

period

[in] Период графика (таймфрейм). Может принимать одно из значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

Возвращаемое значение

При успешном открытии графика функция возвращает идентификатор графика. В противном случае возвращает 0.

Примечание

Максимально возможное количество одновременно открытых графиков в терминале не может быть больше, чем значение CHARTS_MAX.

Пример:

```
#define CHART_SYMBOL  NULL
#define CHART_PERIOD  PERIOD_CURRENT

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- устанавливаем символ и таймфрейм для нового графика
   string symbol=CHART_SYMBOL;
   if(symbol==NULL)
      symbol=Symbol();
   ENUM_TIMEFRAMES timeframe = (CHART_PERIOD==PERIOD_CURRENT ? Period() : CHART_PERIOI

//--- открываем новый график с заданными символом и периодом
   long chart_id=ChartOpen(symbol, timeframe);
   if(chart_id==0)
     {
      Print("ChartOpen() failed. Error ", GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- распечатываем в журнале параметры открытого графика
   PrintFormat("A new chart of the %s symbol has been opened with a period of %s and C
               symbol, StringSubstr(EnumToString(timeframe), 7), chart_id);
   /*
   результат:
   A new chart of the GBPUSD symbol has been opened with a period of M1 and ChartID 13
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartFirst

Возвращает идентификатор первого графика клиентского терминала.

```
long  ChartFirst();
```

Возвращаемое значение

Идентификатор графика.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем идентификатор самого первого графика, открытого в клиентском терминале
   long            chart_id= ChartFirst();

//--- по полученному идентификатору дополнительно получаем символ и период графика
   string          symbol  = ChartSymbol(chart_id);
   ENUM_TIMEFRAMES period  = ChartPeriod(chart_id);

//--- выведем в журнал описание первого графика клиентского терминала
   PrintFormat("ID of the first chart of the client terminal: %I64d, chart symbol: %s,
   /*
   результат:
   ID of the first chart of the client terminal: 133246248352168440, chart symbol: EUR
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartNext

Функция возвращает идентификатор графика, следующего за указанным.

```
long  ChartNext(
   long  chart_id      // идентификатор графика
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 не означает текущий график. 0 означает "вернуть идентификатор первого графика".

Возвращаемое значение

Идентификатор графика. Если список графиков закончился, функция возвращает -1.

Пример:

```
//--- переменные для идентификаторов графиков
   long currChart,prevChart=ChartFirst();
   int i=0,limit=100;
   Print("ChartFirst = ",ChartSymbol(prevChart)," ID = ",prevChart);
   while(i<limit)// у нас наверняка не больше 100 открытых графиков
     {
      currChart=ChartNext(prevChart); // на основании предыдущего получим новый графи
      if(currChart<0) break;          // достигли конца списка графиков
      Print(i,ChartSymbol(currChart)," ID =",currChart);
      prevChart=currChart;// запомним идентификатор текущего графика для ChartNext()
      i++;// не забудем увеличить счетчик
     }
```

© 2000-2025, MetaQuotes Ltd.

ChartClose

Закрывает указанный график.

```
bool  ChartClose(
   long  chart_id=0      // идентификатор графика
   );
```

Параметры

chart_id=0

[in] Идентификатор графика. 0 означает текущий график.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- открываем новый график с символом и периодом как у текущего графика
   long chart_id=ChartOpen(_Symbol, _Period);
   if(chart_id==0)
     {
      Print("ChartOpen() failed. Error ", GetLastError());
      return;
     }

//--- распечатываем в журнале параметры открытого графика
   PrintFormat("A new chart of the %s symbol has been opened with a period of %s and C
               _Symbol, StringSubstr(EnumToString(_Period), 7), chart_id);

//--- ожидаем две секунды и закрываем вновь открытый график
   PrintFormat("Waiting 3 seconds before closing a newly opened chart with ID %I64d ..
   Sleep(3000);
   ResetLastError();
   if(!ChartClose(chart_id))
     {
      Print("ChartClose() failed. Error ", GetLastError());
      return;
     }

//--- распечатываем в журнале сообщение об успешном закрытии графика
   PrintFormat("The chart with ID %I64d was successfully closed", chart_id);
   /*
   результат:
   A new chart of the GBPUSD symbol has been opened with a period of M1 and ChartID 13
```

© 2000-2025, MetaQuotes Ltd.

```
   Waiting 3 seconds before closing a newly opened chart with ID 133346697706632016 ..
   The chart with ID 133346697706632016 was successfully closed
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSymbol

Возвращает имя символа указанного графика.

```
string  ChartSymbol(
   long  chart_id=0      // идентификатор графика
   );
```

Параметры

chart_id=0

[in] Идентификатор графика. 0 означает текущий график.

Возвращаемое значение

Если графика не существует, то возвращается пустая строка.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим символ текущего графика и выведем полученное значение в журнал
   string chart_symbol = ChartSymbol();
   Print("Current chart symbol: ", chart_symbol);

//--- возьмём идентификатор существующего графика (в данном случае текущего)
   long chart_id=ChartID();
   chart_symbol=ChartSymbol(chart_id);
   PrintFormat("Chart symbol with ID %I64d: %s", chart_id, chart_symbol);

//--- зададим случайный идентификатор графика при получении символа
   chart_symbol = ChartSymbol(1234567890);
   if(chart_symbol=="")
      Print("The chart with ID 1234567890 does not exist");
   else
      Print("Chart symbol with ID 1234567890: ", chart_symbol);
   /*
   результат:
   Current chart symbol: GBPUSD
   Chart symbol with ID 132966427583395104: GBPUSD
   The chart with ID 1234567890 does not exist
   */
  }
```

Смотри также

ChartSetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

ChartPeriod

Возвращает значение периода указанного графика.

```
ENUM_TIMEFRAMES  ChartPeriod(
   long  chart_id=0      // идентификатор графика
   );
```

Параметры

chart_id=0

[in] Идентификатор графика. 0 означает текущий график.

Возвращаемое значение

Значение типа ENUM_TIMEFRAMES. Если графика не существует, то возвращается 0.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим период текущего графика и выведем полученное значение в журнал
   ENUM_TIMEFRAMES period=ChartPeriod();
   Print("Current chart period: ", EnumToString(period));

//--- возьмём идентификатор существующего графика (в данном случае текущего)
   long chart_id=ChartID();
   period=ChartPeriod(chart_id);
   PrintFormat("Chart period with ID %I64d: %s", chart_id, EnumToString(period));

//--- зададим случайный идентификатор графика
   period=ChartPeriod(1234567890);
   if(period==0)
      Print("The chart with ID 1234567890 does not exist");
   else
      Print("Chart period with ID 1234567890: ", EnumToString(period));
   /*
   результат:
   Current chart period: PERIOD_M15
   Chart period with ID 133510090240498505: PERIOD_M15
   The chart with ID 1234567890 does not exist
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartRedraw

Вызывает принудительную перерисовку указанного графика.

```
void  ChartRedraw(
   long  chart_id=0      // идентификатор графика
   );
```

Параметры

chart_id=0

[in] Идентификатор графика. 0 означает текущий график.

Примечание

Обычно применяется после изменения свойств объектов.

Пример:

```
#define   WIDTH      50    // ширина прямоугольника в барах

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем идентификатор текущего графика и задаём наименование графического объе
   long   chart_id = ChartID();
   string obj_name = MQLInfoString(MQL_PROGRAM_NAME)+"_RectLabel";

//--- получаем две координаты времени прямоугольника
   datetime time1  = iTime(_Symbol, _Period, WIDTH);
   datetime time2  = iTime(_Symbol, _Period, 0);
   if(time1==0 || time2==0)
     {
      Print("Error getting time ", GetLastError());
      return;
     }

//--- получаем максимальную и минимальную цены в диапазоне ширины прямоугольника
   double price1 = HighestHigh(_Symbol, _Period, 0, WIDTH);
   double price2 = LowestLow(_Symbol, _Period, 0, WIDTH);
   if(price1==EMPTY_VALUE || price2==EMPTY_VALUE)
      return;

//--- создаём объект-прямоугольник
   Print("Create a wheat-colored rectangle");
   if(!ObjectCreate(chart_id, obj_name, OBJ_RECTANGLE, 0, time1, price1, time2, price2
     {
      Print("ObjectCreate() failed. Error ", GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- заполняем прямоугольник изначальным цветом
   ObjectSetInteger(chart_id, obj_name, OBJPROP_FILL, true);
   ObjectSetInteger(chart_id, obj_name, OBJPROP_BACK, true);
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrWheat);
   ChartRedraw();

//--- ждём секунду, заполняем прямоугольник цветом "DodgerBlue" и обновляем график
   Sleep(1000);
   Print("Change color to DodgerBlue");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrDodgerBlue);
   ChartRedraw();

//--- ждём секунду, заполняем прямоугольник цветом "LimeGreen" и обновляем график
   Sleep(1000);
   Print("Change color to LimeGreen");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrLimeGreen);
   ChartRedraw();

//--- ждём секунду, заполняем прямоугольник цветом "OrangeRed" и обновляем график
   Sleep(1000);
   Print("Change color to OrangeRed");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrOrangeRed);
   ChartRedraw();

//--- ждём секунду, заполняем прямоугольник цветом "Wheat" и обновляем график
   Sleep(1000);
   Print("Reset color to original");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrWheat);
   ChartRedraw();

//--- через три секунды удаляем объект с графика
   Sleep(3000);
   Print("Delete the rectangle");
   ObjectDelete(chart_id, obj_name);
  }
//+------------------------------------------------------------------+
//| Возвращает максимальный High в указанном диапазоне баров         |
//+------------------------------------------------------------------+
double HighestHigh(const string symbol, const ENUM_TIMEFRAMES timeframe, const uint st
  {
   ResetLastError();
   int index=iHighest(symbol, timeframe, MODE_HIGH, count, start);
   if(index==-1)
     {
      PrintFormat("%s: iHighest() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   GetLastError();
   double price=iHigh(symbol, timeframe, index);
   if(price==0)
     {
      PrintFormat("%s: iHigh() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   return(price);
  }
//+------------------------------------------------------------------+
//| Возвращает минимальный Low в указанном диапазоне баров           |
//+------------------------------------------------------------------+
double LowestLow(const string symbol, const ENUM_TIMEFRAMES timeframe, const uint sta
  {
   ResetLastError();
   int index=iLowest(symbol, timeframe, MODE_LOW, count, start);
   if(index==-1)
     {
      PrintFormat("%s: iLowest() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   GetLastError();
   double price=iLow(symbol, timeframe, index);
   if(price==0)
     {
      PrintFormat("%s: iLow() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   return(price);
  }
```

Смотри также

Графические объекты

© 2000-2025, MetaQuotes Ltd.

ChartSetDouble

Задает значение соответствующего свойства указанного графика. Свойство графика должно быть типа double. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetDouble(
   long                          chart_id,      // идентификатор графика
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,       // идентификатор свойства
   double                        value          // значение
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

prop_id

[in] Идентификатор свойства графика. Значение может быть одним из значений перечисления ENUM_CHART_PROPERTY_DOUBLE (кроме read-only свойств).

value

[in] Значение свойства.

Возвращаемое значение

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция является асинхронной - это означает, что функция не дожидается выполнения команды, успешно поставленной в очередь указанного графика, а сразу же возвращает управление. Изменение свойства произойдет только после обработки команды в очереди графика. Для немедленного выполнения команд в очереди графика нужно вызвать функцию ChartRedraw.

Если требуется немедленно изменить сразу несколько свойств графика, то необходимо соответствующие функции (ChartSetString, ChartSetDouble, ChartSetString) выполнить в одном блоке кода и затем вызвать один раз ChartRedraw.

Для проверки результата выполнения можно использовать функцию, запрашивающую указанное свойство графика (ChartGetInteger, ChartGetDouble, ChartSetString). При этом необходимо иметь в виду, что данные функции являются синхронными и дожидаются результата выполнения.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем идентификатор текущего графика
   long chart_id = ChartID();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- запоминаем изначальные отступ нулевого бара от правого края в процентах и флаг а
   double shift = ChartGetDouble(chart_id, CHART_SHIFT_SIZE);
   bool   scroll= ChartGetInteger(chart_id,CHART_AUTOSCROLL);

//--- сбрасываем флаг автоматической прокрутки графика
   ChartSetInteger(chart_id, CHART_AUTOSCROLL, true);
   PrintFormat("Initial chart shift size: %.2f", shift);

//--- в цикле от 2.0% до 50.0% с шагом в 0.5%
   for(int i=20; i<=500; i+=5)
     {
      //--- устанавливаем размер отступа как i / 10 и распечатываем в журнале установл
      ChartSetDouble(chart_id, CHART_SHIFT_SIZE, i/10.0);
      PrintFormat("Set chart shift size to %.1f%%", i/10.0);

      //--- удерживаем нулевой бар графика на расстоянии установленного отступа
      ChartNavigate(chart_id, CHART_END, 0);

      //--- немного подождём
      Sleep(16);
     }

//--- устанавливаем изначальные отступ и автопрокрутку графика
   Print("Set the chart shift size to the initial value of ", shift);
   ChartSetDouble(ChartID(), CHART_SHIFT_SIZE, shift);
   ChartSetInteger(chart_id, CHART_AUTOSCROLL, scroll);
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSetInteger

Задает значение соответствующего свойства указанного графика. Свойство графика должно быть типов datetime, int, color, bool или char. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetInteger(
   long                          chart_id,      // идентификатор графика
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // идентификатор свойства
   long                          value          // значение
   );
```

Задает значение соответствующего свойства указанного подокна.

```
bool  ChartSetInteger(
   long                          chart_id,      // идентификатор графика
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // идентификатор свойства
   int                           sub_window,    // номер подокна
   long                          value          // значение
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

prop_id

[in] Идентификатор свойства графика. Значение может быть одним из значений перечисления ENUM_CHART_PROPERTY_INTEGER (кроме read-only свойств).

sub_window

[in] Номер подокна графика. Для первого варианта по умолчанию значение равно 0 (главное окно графика). Большинство свойств не требуют указания номера подокна.

value

[in] Значение свойства.

Возвращаемое значение

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция является асинхронной - это означает, что функция не дожидается выполнения команды, успешно поставленной в очередь указанного графика, а сразу же возвращает управление. Изменение свойства произойдет только после обработки команды в очереди графика. Для немедленного выполнения команд в очереди графика нужно вызвать функцию ChartRedraw.

Если требуется немедленно изменить сразу несколько свойств графика, то необходимо соответствующие функции (ChartSetString, ChartSetDouble, ChartSetString) выполнить в одном блоке кода и затем вызвать один раз ChartRedraw.

© 2000-2025, MetaQuotes Ltd.

Для проверки результата выполнения можно использовать функцию, запрашивающую указанное свойство графика (ChartGetInteger, ChartGetDouble, ChartSetString). При этом необходимо иметь в виду, что данные функции являются синхронными и дожидаются результата выполнения.

Пример:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- включение сообщений о перемещении мыши по окну чарта
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- принудительное обновление свойств графика гарантирует готовность к обработке соб
   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));
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSetString

Задает значение соответствующего свойства указанного графика. Свойство графика должно быть типа string. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetString(
   long                          chart_id,      // идентификатор графика
   ENUM_CHART_PROPERTY_STRING    prop_id,       // идентификатор свойства
   string                        str_value      // значение
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

prop_id

[in] Идентификатор свойства графика. Значение может быть одним из значений перечисления ENUM_CHART_PROPERTY_STRING (кроме read-only свойств).

str_value

[in] Строка для установки свойства. Длина строки не может превышать 2045 символов (лишние символы будут обрезаны).

Возвращаемое значение

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция является асинхронной - это означает, что функция не дожидается выполнения команды, успешно поставленной в очередь указанного графика, а сразу же возвращает управление. Изменение свойства произойдет только после обработки команды в очереди графика. Для немедленного выполнения команд в очереди графика нужно вызвать функцию ChartRedraw.

Если требуется немедленно изменить сразу несколько свойств графика, то необходимо соответствующие функции (ChartSetString, ChartSetDouble, ChartSetString) выполнить в одном блоке кода и затем вызвать один раз ChartRedraw.

Для проверки результата выполнения можно использовать функцию, запрашивающую указанное свойство графика (ChartGetInteger, ChartGetDouble, ChartSetString). При этом необходимо иметь в виду, что данные функции являются синхронными и дожидаются результата выполнения.

Пример:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
```

© 2000-2025, MetaQuotes Ltd.

```
   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);
  }
```

Смотри также

Comment, ChartGetString

© 2000-2025, MetaQuotes Ltd.

ChartGetDouble

Возвращает значение соответствующего свойства указанного графика. Свойство графика должно быть типа double. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
double  ChartGetDouble(
   long                          chart_id,          // идентификатор графика
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,           // идентификатор свойства
   int                           sub_window=0       // номер подокна, если требуется
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  ChartGetDouble(
   long                          chart_id,          // идентификатор графика
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,           // идентификатор свойства
   int                           sub_window,        // номер подокна
   double&                       double_var         // сюда примем значение свойства
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

prop_id

[in] Идентификатор свойства графика. Значение может быть одним из значений перечисления ENUM_CHART_PROPERTY_DOUBLE.

sub_window

[in] Номер подокна графика. Для первого варианта по умолчанию значение равно 0 (главное окно графика). Большинство свойств не требуют указания номера подокна.

double_var

[out] Переменная типа double, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа double.

Для второго варианта вызова возвращает true, если данное свойство поддерживается и значение было помещено в переменную double_var, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция является синхронной - это означает, что она дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом.

Пример:

© 2000-2025, 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartGetInteger

Возвращает значение соответствующего свойства указанного графика. Свойство графика должно быть типов datetime, int или bool. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
long  ChartGetInteger(
   long                          chart_id,      // идентификатор графика
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // идентификатор свойства
   int                           sub_window=0   // номер подокна, если требуется
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  ChartGetInteger(
   long                          chart_id,      // идентификатор графика
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // идентификатор свойства
   int                           sub_window,    // номер подокна
   long&                         long_var       // сюда примем значение свойства
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

prop_id

[in] Идентификатор свойства графика. Значение может быть одним из значений перечисления ENUM_CHART_PROPERTY_INTEGER.

sub_window

[in] Номер подокна графика. Для первого варианта по умолчанию значение равно 0 (главное окно графика). Большинство свойств не требуют указания номера подокна.

long_var

[out] Переменная типа long, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа long.

Для второго варианта вызова возвращает true, если данное свойство поддерживается и значение было помещено в переменную long_var, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция является синхронной - это означает, что она дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом.

Пример:

© 2000-2025, 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");
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartGetString

Возвращает значение соответствующего свойства указанного графика. Свойство графика должно быть типа string. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
string  ChartGetString(
   long  chart_id,                                  // идентификатор графика
   ENUM_CHART_PROPERTY_STRING   prop_id             // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  ChartGetString(
   long    chart_id,                                // идентификатор графика
   ENUM_CHART_PROPERTY_STRING     prop_id,          // идентификатор свойства
   string& string_var                               // сюда примем значение свойства
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

prop_id

[in] Идентификатор свойства графика. Значение может быть одним из значений перечисления ENUM_CHART_PROPERTY_STRING.

string_var

[out] Переменная типа string, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа string.

Для второго варианта вызова возвращает true, если данное свойство поддерживается и значение было помещено в переменную string_var, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция ChartGetString может использоваться для считывания комментариев, выведенных на график функциями Comment или ChartSetString.

Функция является синхронной - это означает, что она дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом.

Пример:

```
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartSetString(0,CHART_COMMENT,"Test comment.\nSecond line.\nThird!");
   ChartRedraw();
   Sleep(1000);
   string comm=ChartGetString(0,CHART_COMMENT);
   Print(comm);
  }
```

Смотри также

Comment, ChartSetString

© 2000-2025, MetaQuotes Ltd.

ChartNavigate

Осуществляет сдвиг указанного графика на указанное количество баров относительно указанной позиции графика.

```
bool  ChartNavigate(
   long                  chart_id,      // идентификатор графика
   ENUM_CHART_POSITION   position,      // позиция
   int                   shift=0        // значение сдвига
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

position

[in] Позиция графика, относительно которой будет произведено смещение. Значение может быть одним из значений перечисления ENUM_CHART_POSITION.

shift=0

[in] Количество баров, на которое необходимо сместить график. Положительное значение означает смещение вправо (к концу графика), отрицательное значение означает смещение влево (к началу графика). Нулевое смещение имеет смысл, когда производится навигация к началу или концу графика.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим handle текущего графика
   long handle=ChartID();
   string comm="";
   if(handle>0) // если получилось, дополнительно настроим
     {
      //--- отключим автопрокрутку
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- установим отступ правого края графика
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- отобразим в виде свечей
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- установить режим отображения тиковых объемов
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);

      //--- подготовим текст для вывода в Comment()
```

© 2000-2025, MetaQuotes Ltd.

```
   comm="Прокрутим на 10 баров вправо от начала истории";
   //--- выведем комментарий
   Comment(comm);
   //--- прокрутим на 10 баров вправо от начала истории
   ChartNavigate(handle,CHART_BEGIN,10);
   //--- получим номер самого первого видимого на графике бара (нумерация как в тай
   long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   //--- добавим символ переноса строки
   comm=comm+"\r\n";
   //--- дополним коментарий
   comm=comm+"Первый бар на графике имеет номер "+IntegerToString(first_bar)+"\r\n"
   //--- выведем комментарий
   Comment(comm);
   //--- подождем 5 секунд, чтобы успеть увидеть, как двигается график
   Sleep(5000);

   //--- допишем текст комментария
   comm=comm+"\r\n"+"Прокрутим на 10 баров влево от правого края графика";
   Comment(comm);
   //--- прокрутим на 10 баров влево от правого края графика
   ChartNavigate(handle,CHART_END,-10);
   //--- получим номер самого первого видимого на графике бара (нумерация как в тай
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"Первый бар на графике имеет номер "+IntegerToString(first_bar)+"\r\n"
   Comment(comm);
   //--- подождем 5 секунд, чтобы успеть увидеть, как двигается график
   Sleep(5000);

   //--- новый блок прокрутки графика
   comm=comm+"\r\n"+"Прокрутим на 300 баров вправо от начала истории";
   Comment(comm);
   //--- прокрутим на 300 баров вправо от начала истории
   ChartNavigate(handle,CHART_BEGIN,300);
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"Первый бар на графике имеет номер "+IntegerToString(first_bar)+"\r\n"
   Comment(comm);
   //--- подождем 5 секунд, чтобы успеть увидеть, как двигается график
   Sleep(5000);

   //--- новый блок прокрутки графика
   comm=comm+"\r\n"+"Прокрутим на 300 баров влево от правого края графика";
   Comment(comm);
   //--- прокрутим на 300 баров влево от правого края графика
   ChartNavigate(handle,CHART_END,-300);
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"Первый бар на графике имеет номер "+IntegerToString(first_bar)+"\r\n"
```

© 2000-2025, MetaQuotes Ltd.

```
      Comment(comm);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartID

Возвращает идентификатор текущего графика.

```
long  ChartID();
```

Возвращаемое значение

Значение типа long.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- переменные для идентификаторов графиков
   long curr_chart=ChartFirst();
   int i=0;
   //--- распечатаем в журнале данные первого графика
   PrintFormat("Chart[%d] ID: %I64d,  symbol: %s", i, curr_chart, ChartSymbol(curr_cha

//--- пока не дошли до лимита открытых графиков (CHARTS_MAX)
   while(!IsStopped() && i < CHARTS_MAX)
     {
      //--- увеличиваем счётчик графиков
      i++;
      //--- на основании предыдущего графика получим идентификатор следующего
      curr_chart=ChartNext(curr_chart);

      //--- если достигли конца списка графиков - завершаем цикл
      if(curr_chart<0)
         break;

      //--- распечатаем в журнале данные очередного графика
      PrintFormat("Chart[%d] ID: %I64d,  symbol: %s", i, curr_chart, ChartSymbol(curr_
     }
   /*
   результат:
   Chart[0] ID: 133246248352168440,  symbol: EURUSD
   Chart[1] ID: 133346697706632015,  symbol: USDJPY
   Chart[2] ID: 133246248352168439,  symbol: GBPUSD
   Chart[3] ID: 133346697706632009,  symbol: RU000A103661
   Chart[4] ID: 133346697706632010,  symbol: AEM4
   Chart[5] ID: 133346697706632011,  symbol: AA.SPB
   Chart[6] ID: 133346697706632012,  symbol: ALLFUTMIX
   Chart[7] ID: 133346697706632013,  symbol: EURUSD
   Chart[8] ID: 133346697706632014,  symbol: SBER
   */
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorAdd

Добавляет на указанное окно графика индикатор с указанным хэндлом. Индикатор и график должны быть построены на одинаковых символе и таймфрейме.

```
bool  ChartIndicatorAdd(
   long  chart_id,                  // идентификатор графика
   int   sub_window                 // номер подокна
   int   indicator_handle           // хэндл индикатора
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

sub_window

[in] Номер подокна графика. 0 означает главное окно графика. Чтобы добавить индикатор в новое окно, параметр должен быть на единицу больше, чем индекс последнего существующего окна, то есть равен CHART_WINDOWS_TOTAL. Если значение параметра превышает значение CHART_WINDOWS_TOTAL, то новое окно создано не будет, индикатор не будет добавлен.

indicator_handle

[in] Хэндл индикатора.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError(). Ошибка 4114 означает, что график и добавляемый индикатор отличаются по символу или таймфрейму.

Примечание

Если на главное окно графика добавляется индикатор, который должен быть отрисован в отдельном подокне (например, встроенный iMACD или пользовательский индикатор с указанным свойством #property indicator_separate_window), то такой индикатор может оказаться невидимым, хотя и будет присутствовать в списке индикаторов. Это означает, что масштаб данного индикатора отличается от масштаба графика цен и значения добавленного индикатора не попадают в отображаемый диапазон ценового графика. При этом GetLastError() вернет нулевой код, означающий отсутствие ошибки. Значения такого "невидимого" индикатора можно будет наблюдать в "Окне данных" и получать из других MQL5-программ.

Пример:

```
#property description "Эксперт для демонстрации работы с функцией ChartIndicatorAdd()."
#property description "После запуска на графике (и получения ошибки в Журнал) откройте"
#property description "свойства советника и задайте правильные параметры <symbol> и <p"
#property description "На график будет добавлен индикатор MACD."

//--- input parameters
input string          symbol="AUDUSD";     // имя символа
input ENUM_TIMEFRAMES period=PERIOD_M12;   // таймфрейм
```

© 2000-2025, MetaQuotes Ltd.

```
input int    fast_ema_period=12;          // период быстрой средней MACD
input int    slow_ema_period=26;          // период медленной средней MACD
input int      signal_period=9;           // период усреднения разности
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // тип цены для расчета MACD

int indicator_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,
//--- попробуем добавить индикатор на график
   if(!AddIndicator())
     {
      //--- функция AddIndicator() отказалась добавить индикатор на график
      int answer=MessageBox("Всё равно попытаться добавить MACD на график?",
                            "Неправильно задан символ и/или таймфрейм для добавления и
                            MB_YESNO // будут показаны кнопки выбора "Yes" и "No"
                            );
      //--- если пользователь все равно настаивает на неправильном использовании Chart
      if(answer==IDYES)
        {
         //--- сначала сообщим об этом в Журнал
         PrintFormat("Внимание! %s: Попробуем добавить индикатор MACD(%s/%s) на графи
                     __FUNCTION__,symbol,EnumToString(period),_Symbol,EnumToString(_Pe
         //--- получим номер нового подокна, в которое попытаемся добавить индикатор
         int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
         //--- теперь сделаем попытку, обреченную на ошибку
         if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
            PrintFormat("Не удалось добавить индикатор MACD на окно %d графика. Код ош
                        subwindow,GetLastError());
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
// эксперт ничего не делает
  }
//+------------------------------------------------------------------+
//| Функция проверки и добавления индикатора на график               |
//+------------------------------------------------------------------+
bool AddIndicator()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- выводимое сообщение
   string message;
//--- проверим на совпадение символ индикатора и символ графика
   if(symbol!=_Symbol)
     {
      message="Демонстрация использования функции Demo_ChartIndicatorAdd():";
      message=message+"\r\n";
      message=message+"Нельзя на график добавить индикатор, рассчитанный на другом сим
      message=message+"\r\n";
      message=message+"Укажите в свойствах эксперта символ графика - "+_Symbol+".";
      Alert(message);
      //--- досрочный выход, не будем добавлять индикатор на график
      return false;
     }
//--- проверим на совпадение таймфрейм индикатора и таймфрейм графика
   if(period!=_Period)
     {
      message="Нельзя на график добавить индикатор, рассчитанный на другом таймфрейме.
      message=message+"\r\n";
      message=message+"Укажите в свойствах эксперта таймфрейм графика - "+EnumToString
      Alert(message);
      //--- досрочный выход, не будем добавлять индикатор на график
      return false;
     }
//--- все проверки прошли, символ и период индикатора соответствуют графику
   if(indicator_handle==INVALID_HANDLE)
     {
      Print(__FUNCTION__,"  Создаем индикатор MACD");
      indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_peri
      if(indicator_handle==INVALID_HANDLE)
        {
         Print("Не удалось создать индикатор MACD. Код ошибки ",GetLastError());
        }
     }
//--- сбросим код ошибки
   ResetLastError();
//--- накладываем индикатор на график
   Print(__FUNCTION__,"  Добавляем индикатор MACD на график");
   Print("MACD построен на ",symbol,"/",EnumToString(period));
//--- получим номер нового подокна, в которое добавим индикатор MACD
   int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   PrintFormat("Добавляем индикатор MACD на окно %d графика",subwindow);
   if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
     {
      PrintFormat("Не удалось добавить индикатор MACD на окно %d графика. Код ошибки
                  subwindow,GetLastError());
     }
//--- добавление индикатора на график прошло успешно
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
```

Смотри также

ChartIndicatorDelete(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorDelete

Удаляет с указанного окна графика индикатор с указанным именем.

```
bool  ChartIndicatorDelete(
   long           chart_id,              // идентификатор графика
   int            sub_window             // номер подокна
   const string   indicator_shortname    // короткое имя индикатора
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

sub_window

[in] Номер подокна графика. 0 означает главное окно графика.

const indicator_shortname

[in] Короткое имя индикатора, которое задается в свойстве INDICATOR_SHORTNAME функцией IndicatorSetString(). Получить короткое имя индикатора можно функцией ChartIndicatorName().

Возвращаемое значение

Возвращает true в случае успешного удаления индикатора, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Если в указанном подокне графика существует несколько индикаторов с одинаковым коротким именем, то будет удален первый по порядку.

Если на значениях удаляемого индикатора построены другие индикаторы на этом же графике, то они также будут удалены.

Не следует путать короткое имя индикатора и имя файла, которое указывается при создании индикатора функциями iCustom() и IndicatorCreate(). Если короткое наименование индикатора не задается явным образом, то при компиляции в нем указывается имя файла, содержащего исходный код индикатора.

Удаление индикатора с графика не означает, что расчетная часть индикатора также будет удалена из памяти терминала. Для освобождения хэндла индикатора используйте функцию IndicatorRelease().

Необходимо правильно формировать короткое имя индикатора, которое с помощью функции IndicatorSetString() записывается в свойство INDICATOR_SHORTNAME. Мы рекомендуем, чтобы короткое имя содержало значения входных параметров индикатора, так как идентификация удаляемого с графика индикатора в функции ChartIndicatorDelete() производится именно по короткому имени.

Пример удаления индикатора при неудачной инициализации:

```
//+------------------------------------------------------------------+
//|                                  Demo_ChartIndicatorDelete.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
```

© 2000-2025, 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
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      first_param=1;
input int      second_param=2;
input int      third_param=3;
input bool     wrong_init=true;
//--- indicator buffers
double         HistogramBuffer[];
string         shortname;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int res=INIT_SUCCEEDED;
//--- привяжем массив HistogramBuffer к индикаторному буферу
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- сконструируем короткое имя индикатора на основе входных параметров
   shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                          first_param,second_param,third_param);
   IndicatorSetString(INDICATOR_SHORTNAME,shortname);
//--- если задано принудительное завершение индикатора, вернем ненулевое значение
   if(wrong_init) res=INIT_FAILED;
   return(res);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const long &volume[],
                const int &spread[])
  {
//--- начальная позиция для работы в цикле
   int start=prev_calculated-1;
   if(start<0) start=0;
//--- заполняем индикаторный буфер значениями
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=close[i];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| обработчик события Deinit                                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   PrintFormat("%s: Код причины деинициализации=%d",__FUNCTION__,reason);
   if(reason==REASON_INITFAILED)
     {
      PrintFormat("Индикатор с коротким именем %s (файл %s) удаляет себя с графика",sh
      int window=ChartWindowFind();
      bool res=ChartIndicatorDelete(0,window,shortname);
      //--- проанализируем результат вызова ChartIndicatorDelete()
      if(!res)
        {
         PrintFormat("Не удалось удалить индикатор %s с окна #%d. Код ошибки %d",
                     shortname,window,GetLastError());
        }
     }
  }
```

Смотри также

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorGet

Возвращает хэндл индикатора с указанным коротким именем на указанном окне графика.

```
int  ChartIndicatorGet(
   long           chart_id,              // идентификатор графика
   int            sub_window             // номер подокна
   const string   indicator_shortname    // короткое имя индикатора
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

sub_window

[in] Номер подокна графика. 0 означает главное окно графика.

const indicator_shortname

[in] Короткое имя индикатора, которое задается в свойстве INDICATOR_SHORTNAME функцией IndicatorSetString(). Получить короткое имя индикатора можно функцией ChartIndicatorName().

Возвращаемое значение

Возвращает хэндл индикатора в случае успешного выполнения, иначе INVALID_HANDLE. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Полученный с помощью функции ChartIndicatorGet() хендл индикатора увеличивает внутренний счетчик использования данного индикатора. Исполняющая система терминала держит загруженными все индикаторы, чей счетчик больше нуля. Поэтому ненужный больше хендл индикатора необходимо сразу и явно освобождать с помощью IndicatorRelease() в той же программе, которая его получила, как это показано в примере ниже. В противном случае найти "брошенный" хендл и освободить его корректно из другой программы не получится.

Необходимо правильно формировать короткое имя индикатора при его создании, которое с помощью функции IndicatorSetString() записывается в свойство INDICATOR_SHORTNAME. Рекомендуется, чтобы короткое имя содержало значения входных параметров индикатора, так как идентификация индикатора в функции ChartIndicatorGet() производится именно по короткому имени.

Другой способ идентификации индикатора - получить список его параметров по заданному хэндлу с помощью функции IndicatorParameters() и затем провести анализ полученных значений.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- количество окон на графике (всегда есть хотя бы одно главное окно)
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- про ходим по окнам
   for(int w=0;w<windows;w++)
     {
      //--- сколько индикаторов в данном окне/подокне
      int total=ChartIndicatorsTotal(0,w);
      //--- переберем все индикаторы в окне
      for(int i=0;i<total;i++)
        {
         //--- получим короткое имя индикатора
         string name=ChartIndicatorName(0,w,i);
         //--- получим хэндл индикатора
         int handle=ChartIndicatorGet(0,w,name);
         //--- выведем в журнал
         PrintFormat("Window=%d,  index=%d,  Name=%s,  handle=%d",w,i,name,handle);
         //--- обязательно освобождаем хендл индикатора, как только он становится нену
         IndicatorRelease(handle);
        }
     }
  }
```

Смотри также

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), IndicatorParameters()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorName

Возвращает короткое имя индикатора по номеру в списке индикаторов на указанном окна графика.

```
string  ChartIndicatorName(
   long   chart_id,       // идентификатор графика
   int    sub_window      // номер подокна
   int    index           // индекс индикатора в списке индикаторов, добавленных к данно
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

sub_window

[in] Номер подокна графика. 0 означает главное окно графика.

index

[in] Индекс индикатора в списке индикаторов. Нумерация индикаторов начинается с нуля, то есть самый первый индикатор в списке имеет нулевой индекс. Количество индикаторов в списке можно получить функцией ChartIndicatorsTotal().

Возвращаемое значение

Короткое имя индикатора, которое задается в свойстве INDICATOR_SHORTNAME функцией IndicatorSetString(). Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Не следует путать короткое имя индикатора и имя файла, которое указывается при создании индикатора функциями iCustom() и IndicatorCreate(). Если короткое наименование индикатора не задается явным образом, то при компиляции в нем указывается имя файла с исходным кодом индикатора.

Необходимо правильно формировать короткое имя индикатора, которое с помощью функции IndicatorSetString() записывается в свойство INDICATOR_SHORTNAME. Мы рекомендуем, чтобы короткое имя содержало значения входных параметров индикатора, так как идентификация удаляемого с графика индикатора в функции ChartIndicatorDelete() производится именно по короткому имени.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем идентификатор текущего графика и количество его подокон, включая главно
   long chart_id = ChartID();
   int  wnd_total= (int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- пройдём в цикле по всем окнам текущего графика
   for(int w=0; w<wnd_total; w++)
     {
      //--- получим количество индикаторов, присоединенных к указанному индексом цикла
      int ind_total=ChartIndicatorsTotal(chart_id, w);

      //--- распечатаем заголовок для выбранного окна графика
      PrintFormat("Chart window %d indicators: ", w);

      //--- получим в цикле и запишем в переменную все имена прикреплённых к выбранном
      string ind_names="";
      for(int i=0; i<ind_total; i++)
        {
         ind_names+="  "+ChartIndicatorName(chart_id, w, i)+(i<ind_total-1 ? "\n": "")
        }
      //--- распечатаем в журнале полученный список имён всех индикаторов, присоединен
      Print(ind_names);
     }
   /*
   результат:
   Chart window 0 indicators:
     AMA(9,2,30)
     SAR(0.02,0.20)
     Fractals
   Chart window 1 indicators:
     RSI(14)
     AMA(9,2,30)
   Chart window 2 indicators:
     MFI(14)
   */
  }
```

Смотри также

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorsTotal

Возвращает количество всех индикаторов, присоединенных к указанному окну графика.

```
int  ChartIndicatorsTotal(
   long  chart_id,       // идентификатор графика
   int   sub_window      // номер подокна
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

sub_window

[in] Номер подокна графика. 0 означает главное окно графика.

Возвращаемое значение

Количество индикаторов на указанном окне графика. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция предназначена для организации перебора всех индикаторов, присоединенных к данному графику. Количества всех окон графика можно получить из свойства CHART_WINDOWS_TOTAL функцией ChartGetInteger().

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем идентификатор текущего графика и количество его подокон, включая главно
   long chart_id = ChartID();
   int  wnd_total= (int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);

//--- пройдём в цикле по всем окнам текущего графика
   for(int w=0; w<wnd_total; w++)
     {
      //--- получим количество индикаторов, присоединенных к указанному индексом цикла
      int ind_total=ChartIndicatorsTotal(chart_id, w);

      //--- распечатаем количество индикаторов, присоединенных к указанному окну графи
      PrintFormat("Chart ID %I64d, subwindow %d. Attached indicators: %d", chart_id, w
     }
   /*
   результат:
   Chart ID 133246248352168439, subwindow 0. Attached indicators: 3
   Chart ID 133246248352168439, subwindow 1. Attached indicators: 2
```

© 2000-2025, MetaQuotes Ltd.

```
   Chart ID 133246248352168439, subwindow 2. Attached indicators: 1
   */
  }
```

Смотри также

ChartIndicatorAdd(), ChartIndicatorDelete(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartWindowOnDropped

Возвращает номер подокна графика, на которое брошен мышкой данный эксперт или скрипт. 0 означает главное окно графика.

```
int  ChartWindowOnDropped();
```

Возвращаемое значение

Значение типа int.

Пример:

```
   int myWindow=ChartWindowOnDropped();
   int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   Print("Скрипт запущен на окне #"+myWindow+
         ". Всего окон на графике "+ChartSymbol()+":",windowsTotal);
```

Смотри также

ChartPriceOnDropped, ChartTimeOnDropped, ChartXOnDropped, ChartYOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartPriceOnDropped

Возвращает ценовую координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

```
double  ChartPriceOnDropped();
```

Возвращаемое значение

Значение типа double.

Пример:

```
   double p=ChartPriceOnDropped();
   Print("ChartPriceOnDropped() = ",p);
```

Смотри также

ChartXOnDropped, ChartYOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartTimeOnDropped

Возвращает временную координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

```
datetime  ChartTimeOnDropped();
```

Возвращаемое значение

Значение типа datetime.

Пример:

```
   datetime t=ChartTimeOnDropped();
   Print("Script wasdropped on the "+t);
```

Смотри также

ChartXOnDropped, ChartYOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartXOnDropped

Возвращает координату по оси X, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

```
int  ChartXOnDropped();
```

Возвращаемое значение

Значение координаты X.

Примечание

Ось X направлена слева направо.

Пример:

```
   int X=ChartXOnDropped();
   int Y=ChartYOnDropped();
   Print("(X,Y) = ("+X+","+Y+")");
```

Смотри также

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartYOnDropped

Возвращает координату по оси Y, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

```
int  ChartYOnDropped();
```

Возвращаемое значение

Значение координаты Y.

Примечание

Ось Y направлена сверху вниз.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем и распечатываем координаты X и Y точки графика, в которую брошен мышкой
   int x=ChartXOnDropped();
   int y=ChartYOnDropped();
   PrintFormat("Script dropped to coordinates X = %d, Y = %d", x, y);
   /*
   результат:
   Script dropped to coordinates X = 429, Y = 114
   */
  }
```

Смотри также

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartSetSymbolPeriod

Меняет значения символа и периода указанного графика. Функция работает асинхронно, то есть отдает команду и не ждет окончания ее выполнения. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetSymbolPeriod(
   long             chart_id,      // идентификатор графика
   string           symbol,        // имя символа
   ENUM_TIMEFRAMES  period         // период
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

symbol

[in] Символ графика. NULL означает символ текущего графика (к которому прикреплен эксперт)

period

[in] Период графика (таймфрейм). Может принимать одно из значений значений перечисления ENUM_TIMEFRAMES. 0 означает период текущего графика.

Возвращаемое значение

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Смена символа/периода влечет за собой переинициализацию эксперта, прикрепленного к соответствующему графику.

Вызов ChartSetSymbolPeriod с тем же символом и таймфреймом можно использовать для обновления графика (аналогично команде Refresh в терминале). Обновление графика в свою очередь запускает перерасчет индикаторов, прикрепленных к нему. Таким образом, вы можете рассчитать индикатор на графике даже при отсутствии тиков (например, в выходные дни).

Пример:

```
#define   SYMBOL    "GBPUSD"
#define   PERIOD    PERIOD_H1

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получаем идентификатор текущего графика, его символ и период
   long            chart_id= ChartID();
   string          symbol  = Symbol();
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES period  = Period();

//--- сообщаем в журнал о смене символа и периода текущего графика на заданные в SYMB
   PrintFormat("Change the %s symbol and the %s period of the chart %I64u to %s %s",
               symbol, TimeframeDescrioption(period), chart_id, SYMBOL, TimeframeDesc
//--- меняем символ и период графика
   ChartSetSymbolPeriod(chart_id, SYMBOL, PERIOD);
   /*
   результат:
   Change the EURUSD symbol and the M1 period of the chart 133246248352168440 to GBPUS
   */
  }
//+------------------------------------------------------------------+
//| Возвращает описание периода графика                              |
//+------------------------------------------------------------------+
string TimeframeDescrioption(const ENUM_TIMEFRAMES timeframe)
  {
   return(StringSubstr(EnumToString(timeframe), 7));
  }
```

Смотри также

ChartSymbol, ChartPeriod

© 2000-2025, MetaQuotes Ltd.

ChartScreenShot

Функция обеспечивает скриншот указанного графика в его текущем состоянии в формате GIF, PNG или BMP в зависимости от указанного расширения.

```
bool  ChartScreenShot(
   long             chart_id,                  // идентификатор графика
   string           filename,                  // имя файла
   int              width,                     // ширина
   int              height,                    // высота
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // тип выравнивания
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

filename

[in] Имя файла скриншота. Не может превышать 63 символов. Скриншот помещается в директорию \Files.

width

[in] Ширина скриншота в пикселях

height

[in] Высота скриншота в пикселях

align_mode=ALIGN_RIGHT

[in] Режим вывода узкого скриншота. Значение перечисления ENUM_ALIGN_MODE. ALIGN_RIGHT означает выравнивание по правой границе (вывод с конца). ALIGN_LEFT задает выравнивание по левой границе.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Если необходимо снять скриншот графика с определенной позиции, то необходимо сначала позиционировать график при помощи функции ChartNavigate(). Если горизонтальный размер скриншота меньше, чем окно графика, то выводится либо правая часть окна графика, либо левая часть, в зависимости от значения параметра align_mode.

Пример:

```
#property description "Советник демонстрирует создание серии скриншотов текущего графи
#property description "с помощью функции ChartScreenShot(). Имя файла для удобства так
#property description "выводится на график. Высота и ширина рисунков задается макросам

#define        WIDTH  800     // ширина рисунка для вызова  ChartScreenShot()
#define        HEIGHT 600     // высота рисунка для вызова  ChartScreenShot()
```

© 2000-2025, MetaQuotes Ltd.

```
//--- input parameters
input int      pictures=5;    // количество рисунков в серии
int            mode=-1;       // -1 означает смещение к правому краю графика, 1 - к ле
int            bars_shift=300;// количество баров при прокрутке графика функцией Chart
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- отключим автопрокрутку графика
   ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- установим отступ правого края графика
   ChartSetInteger(0,CHART_SHIFT,true);
//--- установим отображение графика в виде свечей
   ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
   Print("Подготовка советника к работе завершена");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- вывод имени функции, времени вызова и идентификатора события
   Print(__FUNCTION__,TimeCurrent(),"   id=",id,"   mode=",mode);
//--- обработка события CHARTEVENT_CLICK ("Нажатие кнопки мышки на графике")
   if(id==CHARTEVENT_CLICK)
     {
      //--- начальное смещение от края графика
      int pos=0;
      //--- режим работы с левым краем графика
      if(mode>0)
        {
         //--- прокрутим график к левому краю
         ChartNavigate(0,CHART_BEGIN,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- подготовим подпись на графике и имя для файла
```

© 2000-2025, MetaQuotes Ltd.

```
            string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
            //--- вывод имени на график в виде комментария
            Comment(name);
            //--- сохраним скриншот графика в папку каталог_терминала\MQL5\Files\
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
               Print("Сохранили скриншот ",name);
            //---
            pos+=bars_shift;
            //--- дадим пользователю время чтобы посмотреть на новый участок графика
            Sleep(3000);
            //--- прокрутим график от текущей позиции на bars_shift вправо
            ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
           }
         //--- смена режима на противоположный
         mode*=-1;
        }
      else // режим работы с правым краем графика
        {
         //--- прокрутим график к правому краю
         ChartNavigate(0,CHART_END,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- подготовим подпись на графике и имя для файла
            string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
            //--- вывод имени на график в виде комментария
            Comment(name);
            //--- сохраним скриншот графика в папку каталог_терминала\MQL5\Files\
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
               Print("Сохранили скриншот ",name);
            //---
            pos+=bars_shift;
            //--- дадим пользователю время чтобы посмотреть на новый участок графика
            Sleep(3000);
            //--- прокрутим график от текущей позиции на bars_shift вправо
            ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
           }
         //--- смена режима на противоположный
         mode*=-1;
        }
     }  // конец обработки события CHARTEVENT_CLICK
//--- конец обработчика OnChartEvent()
  }
```

Смотри также

ChartNavigate(), Ресурсы

© 2000-2025, MetaQuotes Ltd.

Торговые функции

Группа функций, предназначенных для управления торговой деятельностью.

Перед тем как приступить к изучению торговых функций платформы, необходимо создать четкое представление об основных терминах: ордер, сделка и позиция:

- Ордер – это распоряжение брокерской компании купить или продать финансовый инструмент. Различают два основных типа ордеров: рыночный и отложенный. Помимо них существуют специальные ордера Тейк Профит и Стоп Лосс.
- Сделка – факт покупки или продажи того или иного финансового инструмента. Покупка (Buy) происходит по цене спроса (Ask), а продажа (Sell) – по цене предложения (Bid). Сделка может быть совершена в результате исполнения рыночного ордера или срабатывания отложенного. Следует учитывать, что в некоторых случаях результатом исполнения ордера могут быть сразу несколько сделок.
- Позиция – это рыночное обязательство, количество купленных или проданных контрактов по финансовому инструменту. Длинная позиция (Long) – купленный в расчете на повышение цены финансовый инструмент, короткая (Short) – обязательство на его поставку в расчете на снижение цены в будущем.

Общая информация о торговых операциях доступна в руководстве пользователя клиентского терминала.

Торговые функции могут использоваться в экспертах и скриптах. Торговые функции могут быть вызваны только в том случае, если в свойствах соответствующего эксперта или скрипта включена галочка "Разрешить советнику торговать".

Разрешение или запрет на торговлю может зависеть от множества факторов, которые описаны в разделе "Разрешение на торговлю".

Функция	Действие
OrderCalcMargin	Вычисляет размер маржи, необходимой для указанного типа ордера, в валюте счета
OrderCalcProfit	Вычисляет размер прибыли на основании переданных параметров в валюте счета
OrderCheck	Проверяет достаточность средств для совершения требуемой торговой операции.
OrderSend	Отправляет торговые запросы на сервер
OrderSendAsync	Отправляет асинхронно торговые запросы без ожидания ответа торгового сервера
PositionsTotal	Возвращает количество открытых позиций
PositionGetSymbol	Возвращает символ соответствующей открытой позиции
PositionSelect	Выбирает открытую позицию для дальнейшей работы с ней
PositionSelectByTicket	Выбирает открытую позицию для дальнейшей работы с ней по указанному тикету

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
PositionGetDouble	Возвращает запрошенное свойство открытой позиции (double)
PositionGetInteger	Возвращает запрошенное свойство открытой позиции (datetime или int)
PositionGetString	Возвращает запрошенное свойство открытой позиции (string)
PositionGetTicket	Возвращает тикет позиции по индексу в списке открытых позиций
OrdersTotal	Возвращает количество ордеров
OrderGetTicket	Возвращает тикет соответствующего ордера
OrderSelect	Выбирает ордер для дальнейшей работы с ним
OrderGetDouble	Возвращает запрошенное свойство ордера (double)
OrderGetInteger	Возвращает запрошенное свойство ордера (datetime или int)
OrderGetString	Возвращает запрошенное свойство ордера (string)
HistorySelect	Запрашивает историю сделок и ордеров за указанный период серверного времени
HistorySelectByPosition	Запрашивает историю сделок и ордеров с указанным идентификатором позиции.
HistoryOrderSelect	Выбирает в истории ордер для дальнейшей работы с ним
HistoryOrdersTotal	Возвращает количество ордеров в истории
HistoryOrderGetTicket	Возвращает тикет соответствующего ордера в истории
HistoryOrderGetDouble	Возвращает запрошенное свойство ордера в истории (double)
HistoryOrderGetInteger	Возвращает запрошенное свойство ордера в истории (datetime или int)
HistoryOrderGetString	Возвращает запрошенное свойство ордера в истории (string)
HistoryDealSelect	Выбирает в истории сделку для дальнейших обращений к ней через соответствующие функции
HistoryDealsTotal	Возвращает количество сделок в истории
HistoryDealGetTicket	Выбирает сделку для дальнейшей обработки и возвращает тикет сделки в истории
HistoryDealGetDouble	Возвращает запрошенное свойство сделки в истории (double)
HistoryDealGetInteger	Возвращает запрошенное свойство сделки в истории (datetime или int)
HistoryDealGetString	Возвращает запрошенное свойство сделки в истории (string)

© 2000-2025, MetaQuotes Ltd.

OrderCalcMargin

Вычисляет размер маржи, необходимой для указанного типа ордера на текущем счете и при текущем рыночном окружении без учета текущих отложенных ордеров и открытых позиций. Позволяет оценить размер маржи для планируемой торговой операции. Значение возвращается в валюте счета.

```
bool  OrderCalcMargin(
   ENUM_ORDER_TYPE       action,           // тип ордера
   string                symbol,           // имя символа
   double                volume,           // объем
   double                price,            // цена открытия
   double&               margin            // переменная для получения значения маржи
   );
```

Параметры

action

[in] Тип ордера, может принимать значения из перечисления ENUM_ORDER_TYPE.

symbol

[in] Имя финансового инструмента.

volume

[in] Объем торговой операции.

price

[in] Цена открытия.

margin

[out] Переменная, в которую будет записан необходимый размер маржи в случае успешного выполнения функции. Вычисление производится как если бы на текущем счете не было отложенных ордеров и открытых позиций. Значение маржи зависит от многих факторов и может меняться при изменении рыночного окружения.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
#define   VOLUME     1.0   // объем ордера
#define   DEVIATION  100   // дистанция установки отложенного ордера
#define   STOP_LIMIT 50    // дистанция StopLimit ордера

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
```

© 2000-2025, MetaQuotes Ltd.

```
   int array_types[8]={ORDER_TYPE_BUY,
                       ORDER_TYPE_SELL,
                       ORDER_TYPE_BUY_LIMIT,
                       ORDER_TYPE_SELL_LIMIT,
                       ORDER_TYPE_BUY_STOP,
                       ORDER_TYPE_SELL_STOP,
                       ORDER_TYPE_BUY_STOP_LIMIT,
                       ORDER_TYPE_SELL_STOP_LIMIT };

//--- в цикле по массиву типов ордеров
   for(int i=0; i<(int)array_types.Size(); i++)
     {
      //--- в зависимости от типа ордера получаем тип ORDER_TYPE_BUY или ORDER_TYPE_SE
      ENUM_ORDER_TYPE type=MarketOrderByOrderType((ENUM_ORDER_TYPE)array_types[i]);

      //--- в зависимости от типа ордера получаем цену
      double price=PriceByOrderType(_Symbol, type);

      //--- получаем размер маржи, необходимой для типа ордера, указанного в action
      double margin=EMPTY_VALUE;
      ResetLastError();
      if(!OrderCalcMargin(type, _Symbol, VOLUME, price, margin))
        {
         Print("OrderCalcMargin() failed. Error ",GetLastError());
         continue;
        }
      //--- распечатываем результат в журнале
      PrintFormat("Margin required for %.2f %s order at price %.*f on %s symbol: %.2f
     }
   /*
   результат:
   Margin required for 1.00 Buy order at price 1.31668 on GBPUSD symbol: 1316.68 USD
   Margin required for 1.00 Sell order at price 1.31661 on GBPUSD symbol: 1316.61 USD
   Margin required for 1.00 Buy Limit order at price 1.31568 on GBPUSD symbol: 1315.68
   Margin required for 1.00 Sell Limit order at price 1.31761 on GBPUSD symbol: 1317.6
   Margin required for 1.00 Buy Stop order at price 1.31768 on GBPUSD symbol: 1317.68
   Margin required for 1.00 Sell Stop order at price 1.31561 on GBPUSD symbol: 1315.61
   Margin required for 1.00 Buy Stop Limit order at price 1.31718 on GBPUSD symbol: 13
   Margin required for 1.00 Sell Stop Limit order at price 1.31611 on GBPUSD symbol: 1
   */
  }
//+------------------------------------------------------------------+
//| Возвращает тип ордера Buy или Sell по типу ордера                |
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE MarketOrderByOrderType(ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY  : case ORDER_TYPE_BUY_LIMIT  : case ORDER_TYPE_BUY_STOP  :
```

© 2000-2025, MetaQuotes Ltd.

```
         return(ORDER_TYPE_BUY);
       case ORDER_TYPE_SELL : case ORDER_TYPE_SELL_LIMIT : case ORDER_TYPE_SELL_STOP :
         return(ORDER_TYPE_SELL);
      }
    return(WRONG_VALUE);
   }
//+------------------------------------------------------------------+
//| Возвращает цену открытия по типу ордера                          |
//+------------------------------------------------------------------+
double PriceByOrderType(const string symbol, const ENUM_ORDER_TYPE order_type)
   {
    int     digits=0;
    double  point=0;
    MqlTick tick={};

//--- получаем значение Point символа
    ResetLastError();
    if(!SymbolInfoDouble(symbol, SYMBOL_POINT, point))
      {
       Print("SymbolInfoDouble() failed. Error ", GetLastError());
       return 0;
      }

//--- получаем значение Digits символа
    long value=0;
    if(!SymbolInfoInteger(symbol, SYMBOL_DIGITS, value))
      {
       Print("SymbolInfoInteger() failed. Error ", GetLastError());
       return 0;
      }
    digits=(int)value;

//--- получаем последние цены по символу
    if(!SymbolInfoTick(symbol, tick))
      {
       Print("SymbolInfoTick() failed. Error ", GetLastError());
       return 0;
      }

//--- в зависимости от типа ордера возвращаем цену
    switch(order_type)
      {
       case ORDER_TYPE_BUY              :  return(tick.ask);
       case ORDER_TYPE_SELL             :  return(tick.bid);
       case ORDER_TYPE_BUY_LIMIT        :  return(NormalizeDouble(tick.ask - DEVIATION
       case ORDER_TYPE_SELL_LIMIT       :  return(NormalizeDouble(tick.bid + DEVIATION
       case ORDER_TYPE_BUY_STOP         :  return(NormalizeDouble(tick.ask + DEVIATION
       case ORDER_TYPE_SELL_STOP        :  return(NormalizeDouble(tick.bid - DEVIATION
       case ORDER_TYPE_BUY_STOP_LIMIT   :  return(NormalizeDouble(tick.ask + DEVIATION
```

© 2000-2025, MetaQuotes Ltd.

```
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return(NormalizeDouble(tick.bid - DEVIATION
      default                          :  return(0);
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE order_type)
  {
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      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");
      case ORDER_TYPE_CLOSE_BY         :  return("Close By");
      default                          :  return("Unknown order type");
     }
  }
//+------------------------------------------------------------------+
```

Смотри также

OrderSend(), Свойства ордеров, Типы торговых операций

© 2000-2025, MetaQuotes Ltd.

OrderCalcProfit

Вычисляет размер прибыли для текущего счета и рыночного окружения на основании переданных параметров. Предназначена для предварительной оценки результата торговой операции. Значение возвращается в валюте счета.

```
bool  OrderCalcProfit(
   ENUM_ORDER_TYPE       action,           // тип ордера (ORDER_TYPE_BUY или ORDER_TYP
   string                symbol,           // имя символа
   double                volume,           // объем
   double                price_open,       // цена открытия
   double                price_close,      // цена закрытия
   double&               profit           // переменная для получения значения прибыли
   );
```

Параметры

action

[in] Тип ордера, может принимать одно из двух значений перечисления ENUM_ORDER_TYPE: ORDER_TYPE_BUY или ORDER_TYPE_SELL.

symbol

[in] Имя финансового инструмента.

volume

[in] Объем торговой операции.

price_open

[in] Цена открытия.

price_close

[in] Цена закрытия.

profit

[out] Переменная, в которую будет записано вычисленное значение прибыли в случае успешного выполнения функции. Значение оценки прибыли зависит от многих факторов и может меняться при изменении рыночного окружения.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Если указан недопустимый тип ордера, функция вернет false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
#define   VOLUME     1.0   // объем ордера
#define   DEVIATION  100   // количество пунктов до цены закрытия

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

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   string currency_profit=SymbolInfoString(_Symbol,SYMBOL_CURRENCY_PROFIT);    // валю
   double close_price=ChartPriceOnDropped(); // ценой закрытия будет ценовая координа

//---
   for(int i=0; i<=ORDER_TYPE_SELL; i++)
     {
      ENUM_ORDER_TYPE order_type=(ENUM_ORDER_TYPE)i;                  // тип ордера: 0
      double open_price=PriceOpenByOrderType(_Symbol, order_type);    // цена открытия

      //--- если цена открытия не получена - продолжаем цикл
      if(open_price==0)
         continue;

      //--- если цена закрытия нулевая (скрипт запущен не перетаскиванием на график)
      if(close_price==0)
         close_price=(order_type==ORDER_TYPE_BUY ? open_price + DEVIATION * _Point : c

      //--- вычислим приблизительный размер прибыли для текущего счета и рыночного окр
      double profit=0;
      ResetLastError();
      if(!OrderCalcProfit(order_type, _Symbol, VOLUME, open_price, close_price, profit
        {
         Print("OrderCalcProfit() failed. Error ", GetLastError());
         continue;
        }

      //--- полученное значение прибыли распечатаем в журнале
      PrintFormat("Estimated profit for %.2f %s position on %s with opening price of %
                  VOLUME, OrderTypeDescription(order_type), _Symbol, _Digits, open_pri

     }
   /*
   результат:
   Estimated profit for 1.00 Buy position on EURUSD with opening price of 1.10757 and
   Estimated profit for 1.00 Sell position on EURUSD with opening price of 1.10753 and
   */
  }
//+------------------------------------------------------------------+
//| Возвращает цену открытия по типу ордера                          |
//+------------------------------------------------------------------+
double PriceOpenByOrderType(const string symbol, const ENUM_ORDER_TYPE order_type)
  {
   MqlTick tick={};

//--- получаем последние цены по символу
   if(!SymbolInfoTick(symbol, tick))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return 0;
     }

//--- в зависимости от типа ордера возвращаем цену
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return(tick.ask);
      case ORDER_TYPE_SELL             :  return(tick.bid);
      default                          :  return(0);
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE order_type)
  {
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      default                          :  return("Unknown order type");
     }
  }
```

Смотри также

OrderSend(), Свойства ордеров, Типы торговых операций

© 2000-2025, MetaQuotes Ltd.

OrderCheck

Функция OrderCheck() проверяет достаточность средств для совершения требуемой торговой операции. Результаты проверки помещаются в поля структуры MqlTradeCheckResult.

```
bool  OrderCheck(
   MqlTradeRequest&        request,      // структура запроса
   MqlTradeCheckResult&    result        // структура ответа
   );
```

Параметры

request

[in] Указатель на структуру типа MqlTradeRequest, которая описывает требуемое торговое действие.

result

[in,out] Указатель на структуру типа MqlTradeCheckResult, в которую будет помещен результат проверки.

Возвращаемое значение

В случае нехватки средств или ошибочно заполненных параметров функция возвращает false. В случае успешной базовой проверки структур (проверка указателей) возвращается true - **это не является свидетельством того, что запрашиваемая торговая операция непременно выполнится успешно**. Для получения подробного описания результата выполнения функции следует анализировать поля структуры *result*.

Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
#define   DEVIATION      5              // допустимое отклонение от цены
#define   VOLUME         1.0            // объем ордера
#define   EXPERT_MAGIC   123            // MagicNumber
#define   DIRECTION      ORDER_TYPE_BUY // направление открываемой позиции (ORDER_TYPE_

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявление и инициализация структур запроса, проверки и результата
   MqlTradeRequest      request={};
   MqlTradeCheckResult check  ={};
   MqlTradeResult       result ={};

//--- подготовим параметры торгового запроса
   PrepareRequest(_Symbol, DIRECTION, VOLUME, request);

//--- проверим параметры торгового запроса
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   bool res=OrderCheck(request, check);
   if(!res)
     {
      PrintFormat("Trade request verification completed with error %d\nServer retcode:
      return;
     }

//--- проверка торгового запроса прошла успешно - выведем в журнал описание полей стру
   Print("Trade request verification completed successfully");
   MqlTradeCheckResultPrint(check, 14);

//--- отправка торгового запроса
   if(!OrderSend(request, result))
      Print("OrderSend error ", GetLastError());    // если отправить запрос не удало

//--- информация об операции
   PrintFormat("Trade request result: retcode=%u, deal=%I64u, order=%I64u", result.ret
   /*
   результат с отключенной автоторговлей в клиентском терминале:
   Trade request verification completed with error 4752
   Server retcode: 10027, comment: AutoTrading disabled by client

   включим автоторговлю и проверим ещё раз на закрытом рынке:
   Experts   automated trading is enabled
   Trade request verification completed successfully
   Retcode:      0
   Balance:      10779.50 USD
   Equity:       10779.50 USD
   Profit:       0.00 USD
   Margin:       1104.79 USD
   Margin free:  9674.71 USD
   Margin level: 975.71 %
   Comment:      Done
   OrderSend error 4756
   Trade request result: retcode=10018, deal=0, order=0

   проверим на открытом рынке:
   Trade request verification completed successfully
   Retcode:      0
   Balance:      10779.50 USD
   Equity:       10779.50 USD
   Profit:       0.00 USD
   Margin:       110.46 USD
   Margin free:  10669.04 USD
   Margin level: 9758.74 %
   Comment:      Done
   Trade request result: retcode=10009, deal=2777010968, order=2802818813
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Подготовка параметров для торгового запроса                      |
//+------------------------------------------------------------------+
void PrepareRequest(const string symbol, const ENUM_ORDER_TYPE order_type, const doubl
  {
   ENUM_ORDER_TYPE type=(DIRECTION !=ORDER_TYPE_BUY ? ORDER_TYPE_SELL : DIRECTION);
   double price=(DIRECTION==ORDER_TYPE_BUY ? SymbolInfoDouble(Symbol(), SYMBOL_ASK) :
//--- параметры запроса
   request.action    = TRADE_ACTION_DEAL; // тип торговой операции
   request.symbol    = symbol;            // символ
   request.volume    = volume;            // объем
   request.type      = type;              // тип ордера
   request.price     = price;             // цена для открытия
   request.deviation = DEVIATION;         // допустимое отклонение от цены
   request.magic     = EXPERT_MAGIC;      // MagicNumber ордера
  }
//+------------------------------------------------------------------+
//| Распечатывает в журнале поля структуры                           |
//| результата проверки торгового запроса                            |
//+------------------------------------------------------------------+
void MqlTradeCheckResultPrint(const MqlTradeCheckResult &check, const uint header_widt
  {
//--- получаем валюту счёта и количество знаков после запятой для валюты счета
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
   int    digits  =(int)AccountInfoInteger(ACCOUNT_CURRENCY_DIGITS);

//--- определяем текст заголовка и ширину поля заголовка
//--- если ширина заголовка передана в функцию равной нулю, то шириной будет размер с
   string header="Retcode:";
   uint w=(header_width==0 ? header.Length()+1 : header_width);
//--- выводим в журнал код возврата с заголовком установленной ширины
   PrintFormat("%-*s%-u", w, header, check.retcode);

//--- выводим в журнал значение баланса, которое будет после выполнения торговой опера
   header="Balance:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.balance, currency);

//--- выводим в журнал значение собственных средств, которое будет после выполнения то
   header="Equity:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.equity, currency);

//--- выводим в журнал значение плавающей прибыли, которое будет после выполнения тор
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.profit, currency);

//--- выводим в журнал размер маржи, необходимый для требуемой торговой операции
```

© 2000-2025, MetaQuotes Ltd.

```
   header="Margin:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.margin, currency);

//--- выводим в журнал размер свободных собственных средств, которые останутся после в
   header="Margin free:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.margin_free, currency);

//--- выводим в журнал уровень маржи, который установится после выполнения  требуемой
   header="Margin level:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.2f %%", w, header, check.margin_level);

//--- выводим в журнал комментарий к коду ответа, описание ошибки
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-s", w, header, check.comment);
  }
```

Смотри также

OrderSend(), Типы торговых операций, Структура торгового запроса, Структура результатов проверки торгового запроса, Структура результата торгового запроса

© 2000-2025, MetaQuotes Ltd.

OrderSend

Функция OrderSend() предназначена для совершения торговых операций через отправку запросов на торговый сервер.

```
bool  OrderSend(
   MqlTradeRequest&  request,      // структура запроса
   MqlTradeResult&   result        // структура ответа
   );
```

Параметры

request

[in] Указатель на структуру типа MqlTradeRequest, описывающую торговое действие клиента.

result

[in,out] Указатель на структуру типа MqlTradeResult, описывающую результат торговой операции в случае успешного выполнения (возврата true).

Возвращаемое значение

В случае успешной базовой проверки структур (проверка указателей) возвращается true - **это не свидетельствует об успешном выполнении торговой операции**. Для получения более подробного описания результата выполнения функции следует анализировать поля структуры *result*.

Примечание

Торговый запрос проходит несколько стадий проверок на торговом сервере. В первую очередь проверяется корректность заполнения всех необходимых полей параметра *request*, и при отсутствии ошибок сервер принимает ордер для дальнейшей обработки. При успешном принятии ордера торговым сервером функция OrderSend() возвращает значение true.

Рекомендуется самостоятельно проверить запрос перед отправкой его торговому серверу. Для проверки запроса существует функция OrderCheck(), которая не только проверит достаточность средств для совершения торговой операции, но и вернет в результатах проверки торгового запроса многие другие полезные параметры:

- код возврата, который сообщит об ошибке в проверяемом запросе;
- значение баланса, которое будет после выполнения торговой операции;
- значение собственных средств, которое будет после выполнения торговой операции;
- значение плавающей прибыли, которое будет после выполнения торговой операции;
- размер маржи, необходимый для требуемой торговой операции;
- размер свободных собственных средств, которые останутся после выполнения требуемой торговой операции;
- уровень маржи, который установится после выполнения требуемой торговой операции;
- комментарий к коду ответа, описание ошибки.

При отправке рыночного ордера (MqlTradeRequest.action=TRADE_ACTION_DEAL) успешный результат функции OrderSend() не означает, что ордер был выполнен (исполнены соответствующие сделки): true в этом случае означает только то, что ордер был успешно размещен в торговой системе для дальнейшего выполнения. Торговый сервер может в возвращаемой структуре результата *result* заполнить значения полей *deal* или *order*, если эти

© 2000-2025, MetaQuotes Ltd.

данные будут ему известны в момент формирования ответа на вызов OrderSend(). В общем случае событие или события исполнения сделок, соответствующих ордеру, могут произойти уже после того, как будет отправлен ответ на вызов OrderSend(). Поэтому для любого типа торгового запроса при получении результата выполнения OrderSend() необходимо в первую очередь проверять код возврата торгового сервера *retcode* и код ответа внешней торговой системы *retcode_external* (при необходимости), которые доступны в возвращаемой структуре результата *result*.

Каждый принятый ордер хранится на торговом сервере в ожидании обработки до тех пор, пока не наступит одно из условий для его исполнения:

- истечение срока действия,
- появление встречного запроса,
- срабатывание ордера при поступлении цены исполнения,
- поступление запроса на отмену ордера.

В момент обработки ордера торговый сервер посылает терминалу сообщение о наступлении торгового события Trade, которое можно обработать функцией OnTrade().

Результат исполнения торгового запроса на сервере, отправленного функцией OrderSend() можно отслеживать при помощи обработчика OnTradeTransaction. При этом следует учитывать, что в результате исполнения одного торгового запроса обработчик OnTradeTransaction будет вызван несколько раз.

Например, при отсылке рыночного ордера на покупку, он обрабатывается, для счета создается соответствующий ордер на покупку, происходит исполнение ордера, его удаление из списка открытых, добавление в историю ордеров, далее добавляется соответствующая сделка в историю и создается новую позицию. Для каждого из эти событий будет вызвана функция OnTradeTransaction.

Пример:

```
//--- значения для ORDER_MAGIC
input long order_magic=55555;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- убедимся,что счет является учебным
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Работа скрипта на реальном счете запрещена!");
      return;
     }
//--- установим либо удалим ордер
   if(GetOrdersTotalByMagic(order_magic)==0)
     {
      //--- текущих ордеров нет - установим ордер
      uint res=SendRandomPendingOrder(order_magic);
      Print("Код возврата торгового сервера ",res);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else // ордера есть - удалим ордера
     {
      DeleteAllOrdersByMagic(order_magic);
     }
//---
  }
//+------------------------------------------------------------------+
//| Получает текущее количество ордеров с указанным ORDER_MAGIC      |
//+------------------------------------------------------------------+
int GetOrdersTotalByMagic(long const magic_number)
  {
   ulong order_ticket;
   int total=0;
//--- пройдем по всем отложенным ордерам
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
   return(total);
  }
//+------------------------------------------------------------------+
//| Удаляет все отложенные ордера с указанным ORDER_MAGIC            |
//+------------------------------------------------------------------+
void DeleteAllOrdersByMagic(long const magic_number)
  {
   ulong order_ticket;
//--- пройдем по всем отложенным ордерам
   for(int i=OrdersTotal()-1;i>=0;i--)
      if((order_ticket=OrderGetTicket(i))>0)
         //--- ордер с подходящим ORDER_MAGIC
         if(magic_number==OrderGetInteger(ORDER_MAGIC))
           {
            MqlTradeResult result={};
            MqlTradeRequest request={};
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- выведем в лог ответ сервера
            Print(__FUNCTION__,": ",result.comment," код ответа ",result.retcode);
           }
//---
  }
//+------------------------------------------------------------------+
//| Установить случайным образом отложенный ордер                    |
//+------------------------------------------------------------------+
uint SendRandomPendingOrder(long const magic_number)
  {
//--- готовим запрос
   MqlTradeRequest request={};
```

© 2000-2025, MetaQuotes Ltd.

```
   request.action=TRADE_ACTION_PENDING;         // установка отложенного ордера
   request.magic=magic_number;                  // ORDER_MAGIC
   request.symbol=_Symbol;                      // инструмент
   request.volume=0.1;                          // объем в 0.1 лот
   request.sl=0;                                // Stop Loss не указан
   request.tp=0;                                // Take Profit не указан
//--- сформируем тип ордера
   request.type=GetRandomType();                // тип ордера
//---сформируем цену для отложенного ордера
   request.price=GetRandomPrice(request.type); // цена для открытия
//--- отправим торговый приказ
   MqlTradeResult result={};
   OrderSend(request,result);
//--- выведем в лог ответ сервера
   Print(__FUNCTION__,":",result.comment);
   if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- вернем код ответа торгового сервера
   return result.retcode;
  }
//+------------------------------------------------------------------+
//| Получить случайным образом тип отложенного типа                  |
//+------------------------------------------------------------------+
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);
     }
//--- недопустимое значение
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------+
//| Получить цену случайным образом                                  |
//+------------------------------------------------------------------+
double GetRandomPrice(ENUM_ORDER_TYPE type)
  {
   int t=(int)type;
//--- уровень стопов по символу
   int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- получим данные последнего тика
   MqlTick last_tick={};
   SymbolInfoTick(_Symbol,last_tick);
//--- вычислим цену в соответствие с типом
   double price;
```

© 2000-2025, MetaQuotes Ltd.

```
   if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT или ORDER_TYPE_SELL_STOP
     {
      price=last_tick.bid; // оттолкнемся от цены Bid
      price=price-(distance+(MathRand()%10)*5)*_Point;
     }
   else             // ORDER_TYPE_SELL_LIMIT или ORDER_TYPE_BUY_STOP
     {
      price=last_tick.ask; // оттолкнемся от цены Ask
      price=price+(distance+(MathRand()%10)*5)*_Point;
     }
//---
   return(price);
  }
```

Смотри также

Типы торговых операций, Структура торгового запроса, Структура результатов проверки торгового запроса, Структура результата торгового запроса

© 2000-2025, MetaQuotes Ltd.

OrderSendAsync

Функция OrderSendAsync() предназначена для проведения асинхронных торговых операций без ожидания ответа торгового сервера на отправленный запрос. Функция предназначена для высокочастотной торговли, когда по условиям торгового алгоритма недопустимо терять время на ожидание ответа от сервера.

```
bool  OrderSendAsync(
   MqlTradeRequest&  request,      // структура запроса
   MqlTradeResult&   result        // структура ответа
   );
```

Параметры

request

[in] Указатель на структуру типа MqlTradeRequest, описывающую торговое действие клиента.

result

[in,out] Указатель на структуру типа MqlTradeResult, описывающую результат торговой операции в случае успешного выполнения функции (при возврате true).

Возвращаемое значение

Возвращает true по факту отправки запроса на торговый сервер. Если запрос не был отправлен, возвращается false. При успешном выполнении функции в переменной *result* код ответа содержит значение TRADE_RETCODE_PLACED (код 10008) - "ордер размещен". Успешное выполнение означает только факт отсылки, но не даёт никакой гарантии, что запрос дошел до торгового сервера и был принят для обработки. Торговый сервер при обработке полученного запроса отправляет клиентскому терминалу ответное сообщение об изменении текущего состояния позиций, ордеров и сделок, которое приводит к генерации события Trade.

Результат исполнения торгового запроса на сервере, отправленного функцией OrderSendAsync() можно отслеживать при помощи обработчика OnTradeTransaction. При этом следует учитывать, что в результате исполнения одного торгового запроса обработчик OnTradeTransaction будет вызван несколько раз.

Например, при отсылке рыночного ордера на покупку, он обрабатывается, для счета создается соответствующий ордер на покупку, происходит исполнение ордера, его удаление из списка открытых, добавление в историю ордеров, далее добавляется соответствующая сделка в историю и создается новую позицию. Для каждого из эти событий будет вызвана функция OnTradeTransaction. Для получения подробной информации следует анализировать параметры этой функции:

- **trans** - в данный параметр передается структура MqlTradeTransaction, описывающая торговую транзакцию, примененную к торговому счету;
- **request** - в данный параметр передается структура MqlTradeRequest, описывающая торговый запрос, в результате которого совершена торговая транзакция;
- **result** - в данный параметр передается структура MqlTradeResult, описывающая результат исполнения торгового запроса.

Примечание

Функция по назначению и параметрам аналогична OrderSend(), но в отличие от неё является асинхронной версией, то есть не приостанавливает работу программы в ожидании результата её

© 2000-2025, MetaQuotes Ltd.

выполнения. Сравнить скорость торговых операций этих двух функций можно с помощью приведенного в примере советника.

Пример:

```
#property description "Эксперт для отправки торговых запросов "
                      " с использованием функции OrderSendAsync().\r\n"
#property description "Показана обработка торговых событий с помощью"
                      " функций-обработчиков OnTrade() и OnTradeTransaction()\r\n"
#property description "В параметрах эксперта можно задать Magic Number"
                      " (уникальный идентификатор) "
#property description "и режим вывода сообщений в журнал "Эксперты". По умолчанию выво
//--- input parameters
input int  MagicNumber=1234567;      // Идентификатор эксперта
input bool DescriptionModeFull=true; // Режим детального вывода
//--- переменная для использования в вызове HistorySelect()
datetime history_start;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- проверим разрешение на автотрейдинг
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
     {
      Alert("Автотрейдинг в терминале запрещен, эксперт будет выгружен.");
      ExpertRemove();
      return(-1);
     }
//--- на реальном счете торговать нельзя
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Советнику запрещено торговать на реальном счете!");
      ExpertRemove();
      return(-2);
     }
//--- можно ли торговать на данном счете (под инвест-паролем нельзя, например)
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
     {
      Alert("Торговля на данном счете запрещена");
      ExpertRemove();
      return(-3);
     }
//--- запомним время запуска эксперта для получения торговой истории
   history_start=TimeCurrent();
//---
   CreateBuySellButtons();
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- удалим за собой графические объекты
   ObjectDelete(0,"Buy");
   ObjectDelete(0,"Sell");
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- заголовок по имени функции-обработчика торгового события
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- получим тип транзакции в виде значения перечисления
   ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
//--- если транзакция является результатом обработки запроса
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      //---выведем название транзакции
      Print(EnumToString(type));
      //--- затем выведем строковое описание обработанного запроса
      Print("------------RequestDescription\r\n",
            RequestDescription(request,DescriptionModeFull));
      //--- и выведем описание результата запроса
      Print("------------ ResultDescription\r\n",
            TradeResultDescription(result,DescriptionModeFull));
     }
   else // для транзакций другого типа выведем полное описание транзакции
     {
      Print("------------ TransactionDescription\r\n",
            TransactionDescription(trans,DescriptionModeFull));
     }
//---
  }
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- статические члены для хранения состояния торгового счета
   static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;
//--- запросим торговую историю
   bool update=HistorySelect(history_start,TimeCurrent());
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("HistorySelect(%s , %s) = %s",
               TimeToString(history_start),TimeToString(TimeCurrent()),(string)update);
//--- заголовок по имени функции-обработчика торгового события
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- выведем имя обработчика и количество ордеров на момент обработки
   int curr_positions=PositionsTotal();
   int curr_orders=OrdersTotal();
   int curr_deals=HistoryOrdersTotal();
   int curr_history_orders=HistoryDealsTotal();
//--- выводим количество ордеров, позиций, сделок, а также изменение в скобках
   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);
//--- вставка разрыва строк для удобного чтения Журнала
   Print("");
//--- запомним состояние счета
   prev_positions=curr_positions;
   prev_orders=curr_orders;
   prev_deals=curr_deals;
   prev_history_orders=curr_history_orders;
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- обработка события CHARTEVENT_CLICK ("Нажатие кнопки мышки на графике")
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("=> ",__FUNCTION__,": sparam = ",sparam);
      //--- минимальный объем для сделки
      double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
      //--- если нажата кнопка "Buy", то покупаем
      if(sparam=="Buy")
        {
         PrintFormat("Buy %s %G lot",_Symbol,volume_min);
         BuyAsync(volume_min);
         //--- отожмем нажатую кнопку обратно
         ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- если нажата кнопка "Sell", то продаем
      if(sparam=="Sell")
        {
         PrintFormat("Sell %s %G lot",_Symbol,volume_min);
         SellAsync(volume_min);
         //--- отожмем нажатую кнопку обратно
         ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
        }
      ChartRedraw();
     }
//---
  }
//+------------------------------------------------------------------+
//| Возвращает текстовое описание транзакции                         |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans,
                              const bool detailed=true)
  {
//--- подготовим строку для возврата из функции
   string desc=EnumToString(trans.type)+"\r\n";
//--- в детальном режиме добавим максимум информации
   if(detailed)
     {
      desc+="Symbol: "+trans.symbol+"\r\n";
      desc+="Deal ticket: "+(string)trans.deal+"\r\n";
      desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
      desc+="Order ticket: "+(string)trans.order+"\r\n";
      desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
      desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
      desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
      desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
      desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
      desc+="Price trigger: "+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";
     }
//--- вернем полученную строку
   return desc;
  }
//+------------------------------------------------------------------+
//| Возвращает текстовое описание торгового запроса                  |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
  {
//--- подготовим строку для возврата из функции
   string desc=EnumToString(request.action)+"\r\n";
//--- в детальном режиме добавим максимум информации
```

© 2000-2025, MetaQuotes Ltd.

```
   if(detailed)
     {
      desc+="Symbol: "+request.symbol+"\r\n";
      desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
      desc+="Order ticket: "+(string)request.order+"\r\n";
      desc+="Order type: "+EnumToString(request.type)+"\r\n";
      desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
      desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
      desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
      desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
      desc+="Deviation points: "+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+="Comment: "+request.comment+"\r\n";
     }
//--- вернем полученную строку
   return desc;
  }
//+------------------------------------------------------------------+
//| Возвращает текстовое описание результата обработки запроса       |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result,
                              const bool detailed=true)
  {
//--- подготовим строку для возврата из функции
   string desc="Retcode "+(string)result.retcode+"\r\n";
//--- в детальном режиме добавим максимум информации
   if(detailed)
     {
      desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
      desc+="Order ticket: "+(string)result.order+"\r\n";
      desc+="Deal ticket: "+(string)result.deal+"\r\n";
      desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
      desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
      desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
      desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
      desc+="Comment: "+result.comment+"\r\n";
     }
//--- вернем полученную строку
   return desc;
  }
//+------------------------------------------------------------------+
//| Создает две кнопки для покупки и продажи                         |
//+------------------------------------------------------------------+
void CreateBuySellButtons()
  {
//--- проверим наличие объекта с именем "Buy"
```

© 2000-2025, MetaQuotes Ltd.

```
   if(ObjectFind(0,"Buy")>=0)
     {
      //--- если найденный объект не является кнопкой, удалим его
      if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Buy");
     }
   else
      ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); // создадим кнопку "Buy"
//--- настроим кнопку "Buy"
   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);
//--- проверим наличие объекта с именем "Sell"
   if(ObjectFind(0,"Sell")>=0)
     {
      //--- если найденный объект не является кнопкой, удалим его
      if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Sell");
     }
   else
      ObjectCreate(0,"Sell",OBJ_BUTTON,0,0,0); // создадим кнопку "Sell"
//--- настроим кнопку "Sell"
   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);
//--- принудительно обновим график, чтобы кнопки отрисовались немедленно
   ChartRedraw();
//---
  }
//+------------------------------------------------------------------+
//| Покупка через асинхронную функцию OrderSendAsync()               |
//+------------------------------------------------------------------+
void BuyAsync(double volume)
  {
//--- подготовим запрос
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_BUY;
```

© 2000-2025, MetaQuotes Ltd.

```
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_ASK);
   req.deviation   =10;
   req.comment     ="Buy using OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": ошибка ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
//| Продажа через асинхронную функцию OrderSendAsync()               |
//+------------------------------------------------------------------+
void SellAsync(double volume)
  {
//--- подготовим запрос
   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     ="Sell using OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": ошибка ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
```

Пример вывода сообщений в журнал "Эксперты":

```
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
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
```

© 2000-2025, MetaQuotes Ltd.

```
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
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
```

© 2000-2025, MetaQuotes Ltd.

```
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
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
```

© 2000-2025, MetaQuotes Ltd.

```
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)
```

© 2000-2025, MetaQuotes Ltd.

PositionsTotal

Возвращает количество открытых позиций.

```
int  PositionsTotal();
```

Возвращаемое значение

Значение типа int.

Примечание

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим и распечатаем в журнале количество открытых позиций на счёте
   int total=PositionsTotal();
   Print("Number of open positions on account: ", total);
   /*
   результат:
   Number of open positions on account: 2
   */
  }
```

Смотри также

PositionGetSymbol(), PositionSelect(), Свойства позиций

© 2000-2025, MetaQuotes Ltd.

PositionGetSymbol

Возвращает символ соответствующей открытой позиции и автоматически выбирает позицию для дальнейшей работы с ней при помощи функций PositionGetDouble, PositionGetInteger, PositionGetString.

```
string  PositionGetSymbol(
   int  index      // номер в списке позиций
   );
```

Параметры

index

[in] Номер позиции в списке открытых позиций.

Возвращаемое значение

Значение типа string. Если позиция не найдена, то вернется пустая строка. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим количество открытых позиций на счёте
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем символ позиции по индексу цикла i
      ResetLastError();
      string symbol=PositionGetSymbol(i);

      //--- если символ позиции успешно получен, то позиция по индексу i автоматически
      //--- и можно получать её свойства при помощи PositionGetDouble, PositionGetInte
      if(symbol!="")
        {
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         PrintFormat("Position symbol at index %d: %s, position type: %s", i, symbol,
```

© 2000-2025, MetaQuotes Ltd.

```
        }
      else
        {
         PrintFormat("PositionGetSymbol(%d) failed. Error %d", i, GetLastError());
         continue;
        }
     }
   /*
   результат:
   Position symbol at index 0: GBPUSD, position type: SELL
   Position symbol at index 1: EURUSD, position type: BUY
   */
  }
```

Смотри также

PositionsTotal(), PositionSelect(), Свойства позиций

© 2000-2025, MetaQuotes Ltd.

PositionSelect

Выбирает открытую позицию для дальнейшей работы с ней. Возвращает true при успешном завершении функции. Возвращает false при неудачном завершении функции. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

```
bool  PositionSelect(
   string  symbol     // имя инструмента
   );
```

Параметры

symbol

[in] Наименование финансового инструмента.

Возвращаемое значение

Значение типа bool.

Примечание

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций. В этом случае, PositionSelect выберет позицию с наименьшим тикетом.

Функция PositionSelect() копирует данные о позиции в программное окружение, и последующие вызовы PositionGetDouble(), PositionGetInteger() и PositionGetString() возвращают ранее скопированные данные. Это означает, что самой позиции может уже и не быть (или же она изменилась по объему, направлению и т.д.), а данные этой позиции можно еще получать. Для гарантированного получения свежих данных о позиции рекомендуется вызывать функцию PositionSelect() непосредственно перед обращением за ними.

Пример:

```
#define   SYMBOL_NAME   "EURUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- выбираем позицию на указанном символе
   if(!PositionSelect(SYMBOL_NAME))
     {
      PrintFormat("PositionSelect(%s) failed. Error %d",SYMBOL_NAME, GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если позиция выбрана, можно получать её данные при помощи PositionGetDouble(), I
//--- получаем тикет выбранной позиции
   ResetLastError();
   long ticket=PositionGetInteger(POSITION_TICKET);
   if(ticket==0)
     {
      PrintFormat("Failed to get %s position ticket. Error %d", SYMBOL_NAME, GetLastEr
      return;
     }

//--- при успешном получении тикета распечатаем символ и тикет выбранной позиции в жур
   PrintFormat("The position that is selected on the %s symbol has ticket %I64d", SYMB
   /*
   результат:
   The position that is selected on the EURUSD symbol has ticket 2810846623
   */
  }
```

Смотри также

PositionGetSymbol(), PositionsTotal(), Свойства позиций

© 2000-2025, MetaQuotes Ltd.

PositionSelectByTicket

Выбирает открытую позицию для дальнейшей работы с ней по указанному тикету. Возвращает true при успешном завершении функции. Возвращает false при неудачном завершении функции. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

```
bool  PositionSelectByTicket(
   ulong    ticket     // тикет позиции
   );
```

Параметры

ticket

[in] Тикет позиции.

Возвращаемое значение

Значение типа bool.

Примечание

Функция PositionSelectByTicket() копирует данные о позиции в программное окружение, и последующие вызовы PositionGetDouble(), PositionGetInteger() и PositionGetString() возвращают ранее скопированные данные. Это означает, что самой позиции может уже и не быть (или же она изменилась по объему, направлению и т.д.), а данные этой позиции можно еще получать. Для гарантированного получения свежих данных о позиции рекомендуется вызывать функцию PositionSelectByTicket() непосредственно перед обращением за ними.

Пример:

```
#define   EXPERT_MAGIC  123456   // MagicNumber

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявляем и инициализируем структуры запроса и результата
   MqlTradeRequest request={};
   MqlTradeResult  result ={};

//--- заполним параметры торгового запроса для открытия длинной позиции
   request.action    = TRADE_ACTION_DEAL;                       // тип торговой операци
   request.symbol    = Symbol();                                // символ
   request.volume    = 0.1;                                     // объем в 0.1 лот
   request.type      = ORDER_TYPE_BUY;                          // тип ордера
   request.price     = SymbolInfoDouble(Symbol(), SYMBOL_ASK); // цена для открытия
   request.deviation = 5;                                       // допустимое отклонени
   request.magic     = EXPERT_MAGIC;                            // MagicNumber ордера

//--- отправляем запрос, если отправить запрос не удалось, выводим код ошибки и заверш
   if(!OrderSend(request, result))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("OrderSend error ", GetLastError());
       return;
      }

//--- выведем информацию об операции
   PrintFormat("Trade request result: retcode: %u, deal: %I64u, order: %I64u", result.

//--- получим тикет позиции из результата торговой операции и выберем позицию по тике
//--- тикет только что открытой позиции соответствует тикету ордера, породившего сдел
   ulong ticket=result.order;
   ResetLastError();
   if(!PositionSelectByTicket(ticket))
     {
      PrintFormat("PositionSelectByTicket(%I64u) failed. Error %d", ticket, GetLastEr
      return;
     }

//--- выведем в журнал данные выбранной по тикету позиции
   ENUM_POSITION_TYPE type  = (ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
   long               time  = PositionGetInteger(POSITION_TIME_MSC);
   double             price = PositionGetDouble(POSITION_PRICE_OPEN);
   double             volume= PositionGetDouble(POSITION_VOLUME);
   string             symbol= PositionGetString(POSITION_SYMBOL);
   int                digits= (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   PrintFormat("Current selected position: %s %.2f %s #%I64u at %.*f, %s",
               symbol, volume, (type==POSITION_TYPE_BUY ? "Buy" : "Sell"), ticket, dig
   /*
   результат:
   Trade request result: retcode: 10009, deal: 2778100901, order: 2803905975
   Current selected position: EURUSD 0.10 Buy #2803905975 at 1.10672, 2024.09.02 12:09
   */
  }
//+------------------------------------------------------------------+
//| Возвращает время с миллисекундами                                |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
  }
```

Смотри также

PositionGetSymbol(), PositionsTotal(), Свойства позиций

© 2000-2025, MetaQuotes Ltd.

PositionGetDouble

Функция возвращает запрошенное свойство открытой позиции, предварительно выбранной при помощи функции PositionGetSymbol или PositionSelect. Свойство позиции должно быть типа double. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
double  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id      // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id,     // идентификатор свойства
   double&                        double_var       // сюда примем значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства позиции. Значение может быть одним из значений перечисления ENUM_POSITION_PROPERTY_DOUBLE.

double_var

[out] Переменная типа double, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа double. В случае неудачного выполнения возвращает 0.

Примечание

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Для гарантированного получения свежих данных о позиции рекомендуется вызывать функцию PositionSelect() непосредственно перед обращением за ними.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- в цикле по всем позициям на счёте
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет очередной позиции, автоматически выбирая позицию для доступ
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип позиции и выводим заголовок для списка вещественных свойств п
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" : "Sell");
      PrintFormat("Double properties of an open position %s #%I64u:", type, ticket);

      //--- распечатываем под заголовком все вещественные свойства выбранной позиции
      PositionPropertiesDoublePrint(15);
     }
   /*
   результат:
   Double properties of an open position Buy #2807075208:
   Volume:        1.00
   Price open:    1.10516
   StopLoss:      0.00000
   TakeProfit:    0.00000
   Price current: 1.10518
   Swap:          0.00
   Profit:        2.00 USD
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал вещественные свойства выбранной позиции         |
//+------------------------------------------------------------------+
void PositionPropertiesDoublePrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   double value=0;

//--- получаем валюту счёта, символ позиции и количество знаков после запятой для симв
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
   string symbol  =PositionGetString(POSITION_SYMBOL);
   int    digits  =(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- определяем текст заголовка и ширину поля заголовка
//--- если ширина заголовка передана в функцию равной нулю, то шириной будет размер ст
   header="Volume:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- получаем и выводим в журнал объём позиции с заголовком установленной ширины
   if(!PositionGetDouble(POSITION_VOLUME, value))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- выводим в журнал значение цены позиции
   header="Price open:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PRICE_OPEN, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- выводим в журнал значение StopLoss
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- выводим в журнал значение TakeProfit
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- выводим в журнал значение Price current
   header="Price current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PRICE_CURRENT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- выводим в журнал значение накопленного свопа
   header="Swap:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_SWAP, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- выводим в журнал значение текущей прибыли
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PROFIT, value))
      return;
   PrintFormat("%-*s%-.2f %s", w, header, value, currency);
  }
```

Смотри также

PositionGetSymbol(), PositionSelect(), Свойства позиций

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

PositionGetInteger

Функция возвращает запрошенное свойство открытой позиции, предварительно выбранной при помощи функции PositionGetSymbol или PositionSelect. Свойство позиции должно быть типа datetime, int. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
long  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id      // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id,     // идентификатор свойства
   long&                           long_var         // сюда примем значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства позиции. Значение может быть одним из значений перечисления ENUM_POSITION_PROPERTY_INTEGER.

long_var

[out] Переменная типа long, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа long. В случае неудачного выполнения возвращает 0.

Примечание

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Для гарантированного получения свежих данных о позиции рекомендуется вызывать функцию PositionSelect() непосредственно перед обращением за ними.

Пример:

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- проверим наличие позиции и выведем время её изменения
   if(PositionSelect(_Symbol))
     {
//--- получим идентификатор позиции для дальнейшей работы с ней
      ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
      Print(_Symbol," position #",position_ID);
//--- получим время образования позиции в миллисекундах с 01.01.1970
      long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
      PrintFormat("Position #%d  POSITION_TIME_MSC = %i64 milliseconds => %s",positior
                  create_time_msc,TimeToString(create_time_msc/1000));
//--- получим время последнего изменения позиции в секундах с 01.01.1970
      long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE = %i64 seconds => %s",
                  position_ID,update_time_sec,TimeToString(update_time_sec));
//--- получим время последнего изменения позиции в миллисекундах с 01.01.1970
      long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE_MSC = %i64 milliseconds => %s",
                  position_ID,update_time_msc,TimeToString(update_time_msc/1000));
     }
//---
  }
```

Смотри также

PositionGetSymbol(), PositionSelect(), Свойства позиций

© 2000-2025, MetaQuotes Ltd.

PositionGetString

Функция возвращает запрошенное свойство открытой позиции, предварительно выбранной при помощи функции PositionGetSymbol или PositionSelect. Свойство позиции должно быть типа string. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
string  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id      // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id,     // идентификатор свойства
   string&                        string_var       // сюда примем значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства позиции. Значение может быть одним из значений перечисления ENUM_POSITION_PROPERTY_STRING.

string_var

[out] Переменная типа string, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа string. В случае неудачного выполнения возвращает пустую строку.

Примечание

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Для гарантированного получения свежих данных о позиции рекомендуется вызывать функцию PositionSelect() непосредственно перед обращением за ними.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- в цикле по всем позициям на счёте
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет очередной позиции, автоматически выбирая позицию для доступ
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип позиции и выводим заголовок для списка строковых свойств пози
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" : "Sell");
      PrintFormat("String properties of an open position %s #%I64u:", type, ticket);

      //--- распечатываем под заголовком все строковые свойства выбранной позиции
      PositionPropertiesStringPrint(15);
     }
   /*
   результат:
   String properties of an open position Buy #2810798881:
   Symbol:         EURUSD
   Comment:        Test PositionGetString
   External ID:
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал строковые свойства выбранной позиции            |
//+------------------------------------------------------------------+
void PositionPropertiesStringPrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value="";

//--- определяем текст заголовка и ширину поля заголовка
//--- если ширина заголовка передана в функцию равной нулю, то шириной будет размер с
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- получаем и выводим в журнал символ позиции с заголовком установленной ширины
   if(!PositionGetString(POSITION_SYMBOL, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- выводим в журнал значение комментария позиции
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetString(POSITION_COMMENT, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выводим в журнал значение идентификатора позиции во внешней системе
   header="External ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetString(POSITION_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
```

Смотри также

PositionGetSymbol(), PositionSelect(), Свойства позиций

© 2000-2025, MetaQuotes Ltd.

PositionGetTicket

Функция возвращает тикет позиции по индексу в списке открытых позиций и автоматически выбирает эту позицию для дальнейшей работы с ней при помощи функций PositionGetDouble, PositionGetInteger, PositionGetString.

```
ulong  PositionGetTicket(
   int  index      // номер в списке позиций
   );
```

Параметры

index

[in] Индекс позиции в списке открытых позиций, начиная с 0.

Возвращаемое значение

Тикет позиции. В случае неудачного выполнения возвращает 0.

Примечание

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Для гарантированного получения свежих данных о позиции рекомендуется вызывать функцию PositionSelect() непосредственно перед обращением за ними.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- в цикле по всем позициям на счёте
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет очередной позиции, автоматически выбирая позицию для доступ
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип позиции и выводим описание выбранной позиции в журнал
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" : "Sell");
      PrintFormat("[%d] Selected position %s #%I64u", i, type, ticket);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   /*
   результат:
   [0] Selected position Sell #2810802718
   [1] Selected position Buy #2810802919
   */
  }
```

Смотри также

PositionGetSymbol(), PositionSelect(), Свойства позиций

© 2000-2025, MetaQuotes Ltd.

OrdersTotal

Возвращает количество действующих ордеров.

```
int  OrdersTotal();
```

Возвращаемое значение

Значение типа int.

Примечание

Не следует путать между собой действующие отложенные ордера и позиции, которые также отображаются на вкладке "Торговля" в панели "Инструменты". Ордер - это распоряжение на проведение торговой операции, а позиция является результатом одной или нескольких сделок.

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим и распечатаем в журнале количество действующих отложенных ордеров на сч
   int total=OrdersTotal();
   Print("Number of active pending orders on the account: ", total);
   /*
   результат:
   Number of active pending orders on the account: 2
   */
  }
```

Смотри также

OrderSelect(), OrderGetTicket(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

OrderGetTicket

Возвращает тикет соответствующего ордера и автоматически выбирает ордер для дальнейшей работы с ним при помощи функций.

```
ulong  OrderGetTicket(
   int  index      // номер в списке ордеров
   );
```

Параметры

index

[in] Номер ордера в списке текущих ордеров.

Возвращаемое значение

Значение типа ulong. В случае неудачного выполнения возвращает 0.

Примечание

Не следует путать между собой действующие отложенные ордера и позиции, которые также отображаются на вкладке "Торговля" в панели "Инструменты". Ордер - это распоряжение на проведение торговой операции, а позиция является результатом одной или нескольких сделок.

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Функция OrderGetTicket() копирует данные об ордере в программное окружение, и последующие вызовы OrderGetDouble(), OrderGetInteger(), OrderGetString() возвращают ранее скопированные данные. Это означает, что самого ордера может уже и не быть (или же в нем изменились цена открытия, уровни Stop Loss / Take Profit или момент истечения), а данные этому ордеру можно еще получать. Для гарантированного получения свежих данных об ордере рекомендуется вызывать функцию OrderGetTicket() непосредственно перед обращением за ними.

Пример:

```
void OnStart()
  {
//--- переменные для получения значений из свойств ордера
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- количество текущих отложенных ордеров
```

© 2000-2025, MetaQuotes Ltd.

```
   uint     total=OrdersTotal();
//--- пройдем в цикле по всем ордерам
   for(uint i=0;i<total;i++)
     {
      //--- получим тикет ордера по его позиции в списке
      if((ticket=OrderGetTicket(i))>0)
        {
         //--- получим свойства ордера
         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)));
         //--- подготовим и выведем информацию об ордере
         printf("#ticket %d %s %G %s at %G was set up at %s",
                ticket,                 // тикет ордера
                type,                   // тип
                initial_volume,         // выставленный объем
                symbol,                 // символ, по которому выставили
                open_price,             // указанная цена открытия
                TimeToString(time_setup)// время установки ордера
                );
        }
     }
//---
  }
```

Смотри также

OrdersTotal(), OrderSelect(), OrderGetInteger()

© 2000-2025, MetaQuotes Ltd.

OrderSelect

Выбирает ордер для дальнейшей работы с ним. Возвращает true при успешном завершении функции. Возвращает false при неудачном завершении функции. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

```
bool  OrderSelect(
   ulong    ticket      // тикет ордера
   );
```

Параметры

ticket

[in] Тикет ордера.

Возвращаемое значение

Значение типа bool.

Примечание

Не следует путать между собой действующие отложенные ордера и позиции, которые также отображаются на вкладке "Торговля" в панели "Инструменты" клиентского терминала.

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Функция OrderSelect() копирует данные об ордере в программное окружение, и последующие вызовы OrderGetDouble(), OrderGetInteger(), OrderGetString() возвращают ранее скопированные данные. Это означает, что самого ордера может уже и не быть (или же в нем изменились цена открытия, уровни Stop Loss / Take Profit или момент истечения), а данные этому ордеру можно еще получать. Для гарантированного получения свежих данных об ордере рекомендуется вызывать функцию OrderSelect() непосредственно перед обращением за ними.

Пример:

```
#define   EXPERT_MAGIC  123456
#define   OFFSET        50                    // отступ от текущей цены для установ
#define   DIRECTION     ORDER_TYPE_BUY_LIMIT  // тип ордера
#define   VOLUME        1.0                   // объём
#define   DEVIATION     2                     // допустимое отклонение от цены

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- объявление и инициализация запроса, результата и переменных
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlTradeRequest request={};
   MqlTradeResult  result ={};
   double          order_price=0;

//--- параметры для установки отложенного ордера
   request.action    = TRADE_ACTION_PENDING;                                  // тип тор
   request.symbol    = _Symbol;                                               // символ
   request.volume    = VOLUME;                                                // объем
   request.deviation = DEVIATION;                                             // допустим
   request.magic     = EXPERT_MAGIC;                                          // MagicNum

//--- проверка типа операции
   switch(DIRECTION)
     {
      case ORDER_TYPE_BUY_LIMIT :
        request.type = ORDER_TYPE_BUY_LIMIT;                                  // тип орде
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_ASK)-OFFSET*_Point;// цена для
        request.price= NormalizeDouble(order_price, _Digits);                 // нормализ
        break;
      case ORDER_TYPE_SELL_LIMIT :
        request.type = ORDER_TYPE_SELL_LIMIT;                                 // тип орде
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_BID)+OFFSET*_Point;// цена для
        request.price= NormalizeDouble(order_price,_Digits);                  // нормализ
        break;
      case ORDER_TYPE_BUY_STOP :
        request.type = ORDER_TYPE_BUY_STOP;                                   // тип орде
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_ASK)+OFFSET*_Point;// цена для
        request.price= NormalizeDouble(order_price,_Digits);                  // нормализ
        break;
      case ORDER_TYPE_SELL_STOP :
        request.type = ORDER_TYPE_SELL_STOP;                                  // тип орде
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_BID)-OFFSET*_Point;// цена для
        request.price= NormalizeDouble(order_price,_Digits);                  // нормализ
        break;
      default: // если выбран не отложенный, или StopLimit-ордер
        Alert("This example is only for placing pending orders BuyLimit, SellLimit, Bu
        break;
     }

//--- отправляем запрос, если отправить запрос не удалось, выводим код ошибки и заверш
   if(!OrderSend(request, result))
     {
      Print("OrderSend error ", GetLastError());
      return;
     }

//--- выведем информацию об операции
   PrintFormat("Trade request result: retcode=%u, order=%I64u", result.retcode, result
```

© 2000-2025, MetaQuotes Ltd.

```
//--- получим тикет ордера из результата торговой операции и выберем ордер по тикету
   ulong ticket=result.order;
   ResetLastError();
   if(!OrderSelect(ticket))
     {
      PrintFormat("OrderSelect(%I64u) failed. Error %d", ticket, GetLastError());
      return;
     }

//--- выведем в журнал данные выбранного по тикету ордера
   ENUM_ORDER_TYPE    type  = (ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE);
   long               time  = OrderGetInteger(ORDER_TIME_SETUP_MSC);
   double             price = OrderGetDouble(ORDER_PRICE_OPEN);
   double             volume= OrderGetDouble(ORDER_VOLUME_CURRENT);
   string             symbol= OrderGetString(ORDER_SYMBOL);
   int                digits= (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   PrintFormat("Current selected order: %s %.2f %s #%I64u at %.*f, %s",
               symbol, volume, OrderTypeDescription(type), ticket, digits, price, Time
   /*
   результат:
   Trade request result: retcode=10009, order=2811006719
   Current selected order: EURUSD 1.00 Buy Limit #2811006719 at 1.10550, 2024.09.04 10
   */
  }
//+------------------------------------------------------------------+
//| Возвращает время с миллисекундами                                |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      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");
      default                          :  return("Unknown order type");
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

OrderGetInteger(), OrderGetDouble(), OrderGetString(), OrderCalcProfit(), OrderGetTicket(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

OrderGetDouble

Возвращает запрошенное свойство ордера, предварительно выбранного при помощи функции OrderGetTicket или OrderSelect. Свойство ордера должно быть типа double. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
double  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,      // идентификатор свойства
   double&                     double_var        // сюда примем значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства ордера. Значение может быть одним из значений перечисления ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] Переменная типа double, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа double. В случае неудачного выполнения возвращает 0.

Примечание

Не следует путать между собой действующие отложенные ордера и позиции, которые также отображаются на вкладке "Торговля" в панели "Инструменты" клиентского терминала.

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Для гарантированного получения свежих данных об ордере рекомендуется вызывать функцию OrderSelect() непосредственно перед обращением за ними.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- в цикле по списку всех ордеров на счёте
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет ордера в списке по индексу цикла
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип ордера и выводим заголовок для списка вещественных свойств вы
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE));
      PrintFormat("Double properties of an active pending order %s #%I64u:", type, tic

      //--- распечатываем под заголовком все вещественные свойства выбранного ордера
      OrderPropertiesDoublePrint(16);
     }
   /*
   результат:
   Double properties of an active pending order Sell Limit #2812000714:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     145.282
   StopLoss:       0.000
   TakeProfit:     0.000
   Price current:  145.044
   StopLimit:      0.000
   Double properties of an active pending order Buy Limit #2812001112:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     144.836
   StopLoss:       0.000
   TakeProfit:     0.000
   Price current:  145.051
   StopLimit:      0.000
   Double properties of an active pending order Buy Stop #2812001488:
   Volume initial: 0.50
   Volume current: 0.50
   Price open:     1.10642
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10530
   StopLimit:      0.00000
   Double properties of an active pending order Sell Stop #2812001712:
   Volume initial: 0.50
   Volume current: 0.50
   Price open:     1.10374
```

© 2000-2025, MetaQuotes Ltd.

```
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10525
   StopLimit:      0.00000
   */
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      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");
      default                          :  return("Unknown order type");
     }
  }
//+------------------------------------------------------------------+
//| Выводит в журнал вещественные свойства выбранного ордера         |
//+------------------------------------------------------------------+
void OrderPropertiesDoublePrint(const uint header_width=0)
  {
//--- получаем символ ордера и количество знаков после запятой для символа
   string symbol = OrderGetString(ORDER_SYMBOL);
   int    digits = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- выводим в журнал первоначальный объём при постановке ордера с заголовком устано
   OrderPropertyPrint("Volume initial:",header_width,2,ORDER_VOLUME_INITIAL);

//--- выводим в журнал значение невыполненного объёма ордера
   OrderPropertyPrint("Volume current:",header_width,2,ORDER_VOLUME_CURRENT);

//--- выводим в журнал значение цены, указанной в ордере
   OrderPropertyPrint("Price open:",header_width,digits,ORDER_PRICE_OPEN);

//--- выводим в журнал значение уровня StopLoss
   OrderPropertyPrint("StopLoss:",header_width,digits,ORDER_SL);

//--- выводим в журнал значение уровня TakeProfit
   OrderPropertyPrint("TakeProfit:",header_width,digits,ORDER_TP);

//--- выводим в журнал значение текущей цены по символу ордера
```

© 2000-2025, MetaQuotes Ltd.

```
   OrderPropertyPrint("Price current:",header_width,digits,ORDER_PRICE_CURRENT);

//--- выводим в журнал значение цены постановки Limit ордера при срабатывании StopLim
   OrderPropertyPrint("StopLimit:",header_width,digits,ORDER_PRICE_STOPLIMIT);
  }
//+------------------------------------------------------------------+
//| Выводит в журнал значение вещественного свойства ордера          |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, int digits, ENUM_ORDE
  {
   double value=0;
   if(!OrderGetDouble(property, value))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property), GetLastE
   else
     {
      //--- если ширина заголовка передана в функцию равной нулю, то шириной будет ра
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-.*f", w, header, digits, value);
     }
  }
```

Смотри также

OrdersTotal(), OrderGetTicket(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

OrderGetInteger

Возвращает запрошенное свойство ордера, предварительно выбранного при помощи функции OrderGetTicket или OrderSelect. Свойство ордера должно быть типа datetime, int. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
long  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id,      // идентификатор свойства
   long&                long_var          // сюда примем значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства ордера. Значение может быть одним из значений перечисления ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] Переменная типа long, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа long. В случае неудачного выполнения возвращает 0.

Примечание

Не следует путать между собой действующие отложенные ордера и позиции, которые также отображаются на вкладке "Торговля" в панели "Инструменты" клиентского терминала.

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Для гарантированного получения свежих данных об ордере рекомендуется вызывать функцию OrderSelect() непосредственно перед обращением за ними.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- в цикле по списку всех ордеров на счёте
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет ордера в списке по индексу цикла
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип ордера и выводим заголовок для списка вещественных свойств вы
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE));
      PrintFormat("Integer properties of an active pending order %s #%I64u:", type, ti

      //--- распечатываем под заголовком все целочисленные свойства выбранного ордера
      OrderPropertiesIntegerPrint(17);
     }
   /*
   результат:
   Integer properties of an active pending order Buy Limit #2812945317:
   Ticket:           2812945317
   Time setup:       2024.09.04 19:17:16
   Type:             Buy Limit
   State:            Placed
   Time expiration:  0
   Time done:        0
   Time setup msc:   2024.09.04 19:17:16.686
   Time done msc:    0
   Type filling:     Return
   Type time:        Time GTC
   Magic:            0
   Reason:           Client
   Position ID:      0
   Position By ID:   0
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал целочисленные свойства выбранного ордера        |
//+------------------------------------------------------------------+
void OrderPropertiesIntegerPrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   long   value=0;

//--- выводим в журнал тикет ордера
   OrderPropertyPrint("Ticket:", header_width, ORDER_TICKET);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выводим в журнал значение времени постановки ордера
   OrderPropertyPrint("Time setup:", header_width, ORDER_TIME_SETUP);

//--- выводим в журнал тип ордера
   OrderPropertyPrint("Type:", header_width, ORDER_TYPE);

//--- выводим в журнал статус ордера
   OrderPropertyPrint("State:", header_width, ORDER_STATE);

//--- выводим в журнал время истечения ордера
   OrderPropertyPrint("Time expiration:", header_width, ORDER_TIME_EXPIRATION);

//--- выводим в журнал время исполнения или снятия ордера
   OrderPropertyPrint("Time done:", header_width, ORDER_TIME_DONE);

//--- выводим в журнал время установки ордера на исполнение в миллисекундах с 01.01.19
   OrderPropertyPrint("Time setup msc:", header_width, ORDER_TIME_SETUP_MSC);

//--- выводим в журнал время исполнения или снятия ордера в миллисекундах с 01.01.197(
   OrderPropertyPrint("Time done msc:", header_width, ORDER_TIME_DONE_MSC);

//--- выводим в журнал тип исполнения по остатку
   OrderPropertyPrint("Type filling:", header_width, ORDER_TYPE_FILLING);

//--- выводим в журнал время жизни ордера
   OrderPropertyPrint("Type time:", header_width, ORDER_TYPE_TIME);

//--- выводим в журнал идентификатор эксперта, выставившего ордер
   OrderPropertyPrint("Magic:", header_width, ORDER_MAGIC);

//--- выводим в журнал причину или источник выставления ордера
   OrderPropertyPrint("Reason:", header_width, ORDER_REASON);

//--- выводим в журнал идентификатор позиции, выставляемый на ордере при его исполнени
   OrderPropertyPrint("Position ID:", header_width, ORDER_POSITION_ID);

//--- выводим в журнал идентификатор встречной позиции для ордеров типа ORDER_TYPE_CLO
   OrderPropertyPrint("Position By ID:", header_width, ORDER_POSITION_BY_ID);
  }
//+------------------------------------------------------------------+
//| Выводит в журнал значение целочисленного свойства ордера         |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, ENUM_ORDER_PROPERTY_IN
  {
   string svalue="";
   long   lvalue=0;
   if(!OrderGetInteger(property, lvalue))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property), GetLastE
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      switch(property)
        {
         case ORDER_TICKET          :
         case ORDER_MAGIC           :
         case ORDER_POSITION_ID     :
         case ORDER_POSITION_BY_ID  :
           svalue=(string)lvalue;
           break;

         case ORDER_TIME_SETUP      :
         case ORDER_TIME_EXPIRATION :
         case ORDER_TIME_DONE       :
           svalue=(lvalue!=0 ? TimeToString((datetime)lvalue, TIME_DATE|TIME_MINUTES|T
           break;

         case ORDER_TIME_SETUP_MSC  :
         case ORDER_TIME_DONE_MSC   :
           svalue=(lvalue!=0 ? TimeMscToString(lvalue) : "0");
           break;

         case ORDER_TYPE            :
           svalue=OrderTypeDescription((ENUM_ORDER_TYPE)lvalue);
           break;
         case ORDER_STATE           :
           svalue=OrderStateDescription((ENUM_ORDER_STATE)lvalue);
           break;
         case ORDER_TYPE_FILLING    :
           svalue=OrderTypeFillingDescription((ENUM_ORDER_TYPE_FILLING)lvalue);
           break;
         case ORDER_TYPE_TIME       :
           svalue=OrderTypeTimeDescription((ENUM_ORDER_TYPE_TIME)lvalue);
           break;
         case ORDER_REASON          :
           svalue=OrderReasonDescription((ENUM_ORDER_REASON)lvalue);
           break;

         default                    :
           svalue="Unknown property";
           break;
        }

      //--- если ширина заголовка передана в функцию равной нулю, то шириной будет раз
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-s", w, header, svalue);
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      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");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание статуса ордера                               |
//+------------------------------------------------------------------+
string OrderStateDescription(ENUM_ORDER_STATE state)
  {
   switch(state)
     {
      case ORDER_STATE_STARTED         :  return("Started");
      case ORDER_STATE_PLACED          :  return("Placed");
      case ORDER_STATE_CANCELED        :  return("Canceled");
      case ORDER_STATE_PARTIAL         :  return("Partial");
      case ORDER_STATE_FILLED          :  return("Filled");
      case ORDER_STATE_REJECTED        :  return("Rejected");
      case ORDER_STATE_EXPIRED         :  return("Expired");
      case ORDER_STATE_REQUEST_ADD     :  return("Request Add");
      case ORDER_STATE_REQUEST_MODIFY  :  return("Request Modify");
      case ORDER_STATE_REQUEST_CANCEL  :  return("Request Cancel");
      default                          :  return("Unknown state: "+(string)state);
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание политики заполнения объема ордера            |
//+------------------------------------------------------------------+
string OrderTypeFillingDescription(const ENUM_ORDER_TYPE_FILLING type)
  {
   switch(type)
     {
      case ORDER_FILLING_FOK     :  return("Fill or Kill");
      case ORDER_FILLING_IOC     :  return("Immediate or Cancel");
      case ORDER_FILLING_BOC     :  return("Book or Cancel");
      case ORDER_FILLING_RETURN  :  return("Return");
      default                    :  return("Unknown type filling: "+(string)type);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
//+------------------------------------------------------------------+
//| Возвращает описание срока действия ордера                        |
//+------------------------------------------------------------------+
string OrderTypeTimeDescription(const ENUM_ORDER_TYPE_TIME type)
   {
    switch(type)
      {
       case ORDER_TIME_GTC           :   return("Time GTC");
       case ORDER_TIME_DAY           :   return("Time Day");
       case ORDER_TIME_SPECIFIED     :   return("Time Specified");
       case ORDER_TIME_SPECIFIED_DAY :   return("Time Specified Day");
       default                       :  return("Unknown type time: "+(string)type);
      }
   }
//+------------------------------------------------------------------+
//| Возвращает описание причины выставления ордера                   |
//+------------------------------------------------------------------+
string OrderReasonDescription(const ENUM_ORDER_REASON reason)
   {
    switch(reason)
      {
       case ORDER_REASON_CLIENT    :  return("Client");
       case ORDER_REASON_MOBILE    :  return("Mobile");
       case ORDER_REASON_WEB       :  return("Web");
       case ORDER_REASON_EXPERT    :  return("Expert");
       case ORDER_REASON_SL        :  return("Stop Loss");
       case ORDER_REASON_TP        :  return("Take Profit");
       case ORDER_REASON_SO        :  return("Stop Out");
       default                     :  return("Unknown reason: "+(string)reason);
      }
   }
//+------------------------------------------------------------------+
//| Возвращает время с миллисекундами                                |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
   {
    return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
   }
```

Смотри также

OrdersTotal(), OrderGetTicket(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

OrderGetString

Возвращает запрошенное свойство ордера, предварительно выбранного при помощи функции OrderGetTicket или OrderSelect. Свойство ордера должно быть типа string. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
string  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id        // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id,      // идентификатор свойства
   string&                     string_var        // сюда примем значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства ордера. Значение может быть одним из значений перечисления ENUM_ORDER_PROPERTY_STRING.

string_var

[out] Переменная типа string, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа string.

Примечание

Не следует путать между собой действующие отложенные ордера и позиции, которые также отображаются на вкладке "Торговля" в панели "Инструменты" клиентского терминала.

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций.

Для гарантированного получения свежих данных об ордере рекомендуется вызывать функцию OrderSelect() непосредственно перед обращением за ними.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- в цикле по списку всех ордеров на счёте
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет ордера в списке по индексу цикла
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип ордера и выводим заголовок для списка строковых свойств выбра
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE));
      PrintFormat("String properties of an active pending order %s #%I64u:", type, tic

      //--- распечатываем под заголовком все строковые свойства выбранного ордера
      OrderPropertiesStringPrint(13);
     }
   /*
   результат:
   String properties of an active pending order Sell Limit #2813781342:
   Comment:     Test OrderGetString
   Symbol:      EURUSD
   External ID:
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал строковые свойства выбранного ордера            |
//+------------------------------------------------------------------+
void OrderPropertiesStringPrint(const uint header_width=0)
  {
//--- выводим в журнал комментарий
   OrderPropertyPrint("Comment:", header_width, ORDER_COMMENT);

//--- выводим в журнал символ, по которому выставлен ордер
   OrderPropertyPrint("Symbol:", header_width, ORDER_SYMBOL);

//--- выводим в журнал идентификатор ордера во внешней системе
   OrderPropertyPrint("External ID:", header_width, ORDER_EXTERNAL_ID);
  }
//+------------------------------------------------------------------+
//| Выводит в журнал значение строкового свойства ордера             |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, ENUM_ORDER_PROPERTY_ST
  {
   string value="";
   if(!OrderGetString(property, value))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property), GetLastE
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- если ширина заголовка передана в функцию равной нулю, то шириной будет раз
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-s", w, header, value);
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      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");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
```

Смотри также

OrdersTotal(), OrderGetTicket(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

HistorySelect

Запрашивает историю сделок и ордеров за указанный период серверного времени.

```
bool  HistorySelect(
   datetime  from_date,      // с даты
   datetime  to_date         // по дату
   );
```

Параметры

from_date

[in] Начальная дата запроса.

to_date

[in] Конечная дата запроса.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Функция HistorySelect() создает в mql5-программе список ордеров и список сделок для дальнейшего обращения к элементам списка посредством соответствующих функций. Размер списка сделок можно узнать с помощью функции HistoryDealsTotal(), размер списка ордеров в истории можно получить с HistoryOrdersTotal(). Перебор элементов списка ордеров лучше всего проводить функцией HistoryOrderGetTicket(), для элементов списка сделок соответственно подходит функция HistoryDealGetTicket().

После применения функции HistoryOrderSelect() список ордеров в истории, доступных mql5-программе, сбрасывается и заполняется заново найденным ордером, если поиск ордера по тикету завершился успешно. То же самое относится к списку сделок, доступных mql5-программе - он сбрасывается функцией HistoryDealSelect() и заполняется заново в случае успешного получения сделки по номеру тикета.

Пример:

```
void OnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- request trade history
   HistorySelect(0,TimeCurrent());
//--- create objects
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
   long     type;
```

© 2000-2025, MetaQuotes Ltd.

```
   long     entry;
//--- for all deals
   for(uint i=0;i<total;i++)
     {
      //--- try to get deals ticket
      if((ticket=HistoryDealGetTicket(i))>0)
        {
         //--- get deals properties
         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);
         //--- only for current symbol
         if(price && time && symbol==Symbol())
           {
            //--- create price object
            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);
            //--- set object properties
            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
           }
        }
     }
//--- apply on chart
   ChartRedraw();
  }
```

Смотри также

HistoryOrderSelect(), HistoryDealSelect()

© 2000-2025, MetaQuotes Ltd.

HistorySelectByPosition

Запрашивает историю сделок и ордеров, имеющих указанный идентификатор позиции.

```
bool  HistorySelectByPosition(
   ulong   position_id     // идентификатор позиции - POSITION_IDENTIFIER
   );
```

Параметры

position_id

[in] Идентификатор позиции, который проставляется на каждом исполненном ордере и на каждой сделке.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Не следует путать между собой ордера из торговой истории и действующие отложенные ордера, которые отображаются на вкладке "Торговля" в панели "Инструменты". Список ордеров, которые были отменены или привели к проведению торговой операции, можно посмотреть в закладке "История" на панели "Инструменты" клиентского терминала.

Функция HistorySelectByPosition() создает в mql5-программе список ордеров и список сделок с указанным идентификатором позиции для дальнейшего обращения к элементам списка посредством соответствующих функций. Размер списка сделок можно узнать с помощью функции HistoryDealsTotal(), размер списка ордеров в истории можно получить с HistoryOrdersTotal(). Перебор элементов списка ордеров лучше всего проводить функцией HistoryOrderGetTicket(), для элементов списка сделок соответственно подходит функция HistoryDealGetTicket().

После применения функции HistoryOrderSelect() список ордеров в истории, доступных mql5-программе, сбрасывается и заполняется заново найденным ордером, если поиск ордера по тикету завершился успешно. То же самое относится к списку сделок, доступных mql5-программе - он сбрасывается функцией HistoryDealSelect() и заполняется заново в случае успешного получения сделки по номеру тикета.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   long  pos_id_array[];          // массив для хранения идентификаторов позиций

//--- запрашиваем всю историю
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- соберём в массив все Position ID, возникшие только от отложенных ордеров
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket, ORDER_TYPE)
      long pos_id=HistoryOrderGetInteger(ticket, ORDER_POSITION_ID);
      if(type<=ORDER_TYPE_SELL || pos_id==0)
         continue;

      int size=ArraySize(pos_id_array);
      if(ArrayResize(pos_id_array, size+1)==size+1)
         pos_id_array[size]=pos_id;
     }

//--- по списку идентификаторов позиций в массиве
   total=ArraySize(pos_id_array);
   for(int i=0; i<total; i++)
     {
      //--- распечатываем в журнале заголовок и список ордеров и сделок позиции
      long position_id=pos_id_array[i];
      Print("List of orders and deals for position with ID: ", position_id);
      HistorySelectByPositionProcess(position_id);
     }
   /*
   результат:
   List of orders and deals for position with ID: 1819629924
     [0] Order Sell Limit #1819629924
     [1] Order Buy #1819633194
     [0] Entry In Deal Sell #1794972472
     [1] Entry Out Deal Buy #1794975589
   List of orders and deals for position with ID: 1841753970
     [0] Order Sell Stop #1841753970
     [1] Order Buy #1842322160
     [0] Entry In Deal Sell #1817242142
     [1] Entry Out Deal Buy #1817765341
   */
  }
//+------------------------------------------------------------------+
//| Выбирает историю ордеров и сделок по идентификатору позиции и    |
//| распечатывает в журнале список ордеров и сделок этой позиции     |
//+------------------------------------------------------------------+
bool HistorySelectByPositionProcess(const long position_id)
  {
//--- запрашиваем историю сделок и ордеров, имеющих указанный идентификатор позиции
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!HistorySelectByPosition(position_id))
     {
      PrintFormat("HistorySelectByPosition(%I64d) failed. Error %d", position_id, GetL
      return(false);
     }

//--- распечатаем в журнале список ордеров позиции
   int orders_total=HistoryOrdersTotal();
   for(int i=0; i<orders_total; i++)
     {
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_ORDER_TYPE order_type=(ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket, ORDEF
      PrintFormat("  [%d] Order %s #%I64u", i, OrderTypeDescription(order_type), ticke
     }

//--- распечатаем в журнале список сделок позиции
   int deals_total =HistoryDealsTotal();
   for(int i=0; i<deals_total; i++)
     {
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_DEAL_ENTRY deal_entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket, DEAL_E
      ENUM_DEAL_TYPE  deal_type= (ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket, DEAL_TY
      if(deal_type!=DEAL_TYPE_BUY && deal_type!=DEAL_TYPE_SELL)
         continue;
      PrintFormat("  [%d] Entry %s Deal %s #%I64u", i, DealEntryDescription(deal_entry
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      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");
      default                          :  return("Unknown order type: "+(string)type);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа сделки позиции                          |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      //--- вернём описание типов сделок только Buy и Sell,
      //--- так как все остальные типы к позиции не относятся
      case DEAL_TYPE_BUY    :  return("Buy");
      case DEAL_TYPE_SELL   :  return("Sell");
      default               :  return("Unknown deal type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание способа изменения позиции                    |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("InOut");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

Смотри также

HistorySelect(), HistoryOrderGetTicket(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

HistoryOrderSelect

Выбирает в истории ордер для последующих обращений к нему через соответствующие функции. Возвращает true при успешном завершении функции. Возвращает false при неудачном завершении функции. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

```
bool  HistoryOrderSelect(
   ulong  ticket      // тикет ордера
   );
```

Параметры

ticket

[in] Тикет ордера

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Не следует путать между собой ордера из торговой истории и действующие отложенные ордера, которые отображаются на вкладке "Торговля" в панели "Инструменты". Список ордеров, которые были отменены или привели к проведению торговой операции, можно посмотреть в закладке "История" на панели "Инструменты" клиентского терминала.

Функция HistoryOrderSelect() очищает в mql5-программе список ордеров из истории, доступных для обращений, и копирует в него один единственный ордер, если выполнение HistoryOrderSelect() завершилось успешно. Если необходимо перебрать все ордеры, выбранные функцией HistorySelect(), то лучше использовать функцию HistoryOrderGetTicket().

Пример:

```
#define   TICKET    1819621374   // тикет любого известного ордера, например, из истор
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- выбираем исторический ордер по тикету, указанному в TICKET
   if(!HistoryOrderSelect(TICKET))
     {
      PrintFormat("HistoryOrderSelect(%I64u) failed. Error %d", TICKET, GetLastError()
      return;
     }

//--- если ордер успешно выбран, получаем его данные и выводим описание ордера в журна
   ENUM_ORDER_TYPE   order_type  = (ENUM_ORDER_TYPE)HistoryOrderGetInteger(TICKET, ORI
   ENUM_ORDER_STATE  order_state = (ENUM_ORDER_STATE)HistoryOrderGetInteger(TICKET, OR
   ENUM_ORDER_REASON order_reason= (ENUM_ORDER_REASON)HistoryOrderGetInteger(TICKET, O
   long              order_time  = HistoryOrderGetInteger(TICKET, ORDER_TIME_SETUP_MSC
   string            order_symbol=HistoryOrderGetString(TICKET, ORDER_SYMBOL);
```

© 2000-2025, MetaQuotes Ltd.

```
   double            order_vol_init=HistoryOrderGetDouble(TICKET, ORDER_VOLUME_INITIAL
   double            order_vol_curr=HistoryOrderGetDouble(TICKET, ORDER_VOLUME_CURRENT
   PrintFormat("%s Order %.2f/%s %s #%I64u %s by %s at %s",
               order_symbol, order_vol_init, (order_vol_curr>0 ? DoubleToString(order_
               OrderTypeDescription(order_type), TICKET, OrderStateDescription(order_s
               OrderReasonDescription(order_reason), TimeMscToString(order_time));
   /*
   результат для различных указанных тикетов:
   EURUSD Order 0.50/0.50 Buy Limit #2812894647 Canceled by Client at 2024.09.04 19:02
   EURUSD Order 0.10/0 Sell #1753011743 Filled by Take Profit at 2023.06.12 17:04:20.3
   GBPUSD Order 0.10/0 Buy #1819621374 Filled by Client at 2023.07.24 06:16:25.746
   */
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      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");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание статуса ордера                               |
//+------------------------------------------------------------------+
string OrderStateDescription(ENUM_ORDER_STATE state)
  {
   switch(state)
     {
      case ORDER_STATE_STARTED         :  return("Started");
      case ORDER_STATE_PLACED          :  return("Placed");
      case ORDER_STATE_CANCELED        :  return("Canceled");
      case ORDER_STATE_PARTIAL         :  return("Partial");
      case ORDER_STATE_FILLED          :  return("Filled");
      case ORDER_STATE_REJECTED        :  return("Rejected");
      case ORDER_STATE_EXPIRED         :  return("Expired");
      case ORDER_STATE_REQUEST_ADD     :  return("Request Add");
      case ORDER_STATE_REQUEST_MODIFY  :  return("Request Modify");
      case ORDER_STATE_REQUEST_CANCEL  :  return("Request Cancel");
      default                          :  return("Unknown state: "+(string)state);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
//+------------------------------------------------------------------+
//| Возвращает описание причины выставления ордера                   |
//+------------------------------------------------------------------+
string OrderReasonDescription(const ENUM_ORDER_REASON reason)
   {
    switch(reason)
      {
       case ORDER_REASON_CLIENT    :  return("Client");
       case ORDER_REASON_MOBILE    :  return("Mobile");
       case ORDER_REASON_WEB       :  return("Web");
       case ORDER_REASON_EXPERT    :  return("Expert");
       case ORDER_REASON_SL        :  return("Stop Loss");
       case ORDER_REASON_TP        :  return("Take Profit");
       case ORDER_REASON_SO        :  return("Stop Out");
       default                     :  return("Unknown reason: "+(string)reason);
      }
   }
//+------------------------------------------------------------------+
//| Возвращает время с миллисекундами                                |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
   {
    return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
   }
```

Смотри также

HistorySelect(), HistoryOrderGetTicket(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

HistoryOrdersTotal

Возвращает количество ордеров в истории. Перед вызовом функции HistoryOrdersTotal() необходимо получить историю сделок и ордеров с помощью функции HistorySelect() или HistorySelectByPosition().

```
int  HistoryOrdersTotal();
```

Возвращаемое значение

Значение типа int.

Примечание

Не следует путать между собой ордера из торговой истории и действующие отложенные ордера, которые отображаются на вкладке "Торговля" в панели "Инструменты". Список ордеров, которые были отменены или привели к проведению торговой операции, можно посмотреть в закладке "История" на панели "Инструменты" клиентского терминала.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- запрашиваем всю имеющуюся историю на счёте
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- получаем количество ордеров в списке и выводим значение в журнал
   int total=HistoryOrdersTotal();
   Print("Number of historical orders on the account: ", total);
   /*
   результат:
   Number of historical orders on the account: 496
   */
  }
```

Смотри также

HistorySelect(), HistoryOrderSelect(), HistoryOrderGetTicket(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetTicket

Возвращает тикет соответствующего ордера в истории. Перед вызовом функции HistoryOrderGetTicket() необходимо получить историю сделок и ордеров с помощью функции HistorySelect() или HistorySelectByPosition().

```
ulong  HistoryOrderGetTicket(
   int  index      // номер в списке ордеров
   );
```

Параметры

index

[in] Номер ордера в списке ордеров.

Возвращаемое значение

Значение типа ulong. В случае неудачного выполнения возвращает 0.

Примечание

Не следует путать между собой ордера из торговой истории и действующие отложенные ордера, которые отображаются на вкладке "Торговля" в панели "Инструменты". Список ордеров, которые были отменены или привели к проведению торговой операции, можно посмотреть в закладке "История" на панели "Инструменты" клиентского терминала.

Пример:

```
void OnStart()
  {
   datetime from=0;
   datetime to=TimeCurrent();
//--- запросить всю историю
   HistorySelect(from,to);
//--- переменные для получения значений из свойств ордера
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   datetime time_done;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- количество текущих отложенных ордеров
   uint     total=HistoryOrdersTotal();
//--- пройдем в цикле по всем ордерам
   for(uint i=0;i<total;i++)
     {
      //--- получим тикет ордера по его позиции в списке
      if((ticket=HistoryOrderGetTicket(i))>0)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- получим свойства ордера
         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));
         //--- подготовим и выведм информацию об ордере
         printf("#ticket %d %s %G %s at %G was set up at %s => done at %s, pos ID=%d",
                ticket,                   // тикет ордера
                type,                     // тип
                initial_volume,           // выставленный объем
                symbol,                   // символ, по которому выставили
                open_price,               // указанная цена открытия
                TimeToString(time_setup),// время установки ордера
                TimeToString(time_done), // время исполнения илм удаления
                positionID                // ID позиции, в которую влилась сделка по ор
                );
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Возвращает строковое наименование типа ордера                    |
//+------------------------------------------------------------------+
string GetOrderType(long type)
  {
   string str_type="unknown operation";
   switch(type)
     {
      case (ORDER_TYPE_BUY):             return("buy");
      case (ORDER_TYPE_SELL):            return("sell");
      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);
  }
```

Смотри также

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetDouble

Возвращает запрошенное свойство ордера. Свойство ордера должно быть типа double. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
double  HistoryOrderGetDouble(
   ulong                      ticket_number,     // тикет
   ENUM_ORDER_PROPERTY_DOUBLE  property_id       // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  HistoryOrderGetDouble(
   ulong                      ticket_number,     // тикет
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,      // идентификатор свойства
   double&                    double_var         // сюда примем значение свойства
   );
```

Параметры

ticket_number

[in] Тикет ордера.

property_id

[in] Идентификатор свойства ордера. Значение может быть одним из значений перечисления ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] Переменная типа double, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа double.

Примечание

Не следует путать между собой ордера из торговой истории и действующие отложенные ордера, которые отображаются на вкладке "Торговля" в панели "Инструменты". Список ордеров, которые были отменены или привели к проведению торговой операции, можно посмотреть в закладке "История" на панели "Инструменты" клиентского терминала.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- запрашиваем историю сделок и ордеров
   if(!HistorySelect(0, TimeCurrent()))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("HistorySelect() failed. Error ", GetLastError());
       return;
      }

//--- в цикле по списку всех исторических ордеров на счёте
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет ордера в списке по индексу цикла
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип ордера и выводим заголовок для списка вещественных свойств вы
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket,
      PrintFormat("Double properties of an history order %s #%I64u:", type, ticket);

      //--- распечатываем под заголовком все вещественные свойства выбранного ордера
      HistoryOrderPropertiesDoublePrint(ticket, 16);
     }
   /*
   результат:
   Double properties of an history order Sell #2810847541:
   Volume initial: 0.50
   Volume current: 0.00
   Price open:     1.10491
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10491
   StopLimit:      0.00000
   Double properties of an history order Buy Limit #2811003507:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     1.10547
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10591
   StopLimit:      0.00000
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал вещественные свойства                           |
//| выбранного исторического ордера                                  |
//+------------------------------------------------------------------+
void HistoryOrderPropertiesDoublePrint(const long ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
```

© 2000-2025, MetaQuotes Ltd.

```
   double value=0;

//--- получаем символ ордера и количество знаков после запятой для символа
   string symbol = HistoryOrderGetString(ticket, ORDER_SYMBOL);
   int    digits = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- определяем текст заголовка и ширину поля заголовка
//--- если ширина заголовка передана в функцию равной нулю, то шириной будет размер ст
   header="Volume initial:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- получаем и выводим в журнал первоначальный объём при постановке ордера с заголов
   if(!HistoryOrderGetDouble(ticket, ORDER_VOLUME_INITIAL, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- выводим в журнал значение невыполненного объёма ордера
   header="Volume current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_VOLUME_CURRENT, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- выводим в журнал значение цены, указанной в ордере
   header="Price open:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_OPEN, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- выводим в журнал значение уровня StopLoss
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- выводим в журнал значение уровня TakeProfit
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- выводим в журнал значение текущей цены по символу ордера
   header="Price current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_CURRENT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выводим в журнал значение цены постановки Limit ордера при срабатывании StopLimi
   header="StopLimit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_STOPLIMIT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      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");
      default                          :  return("Unknown order type");
     }
  }
```

Смотри также

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetInteger

Возвращает запрошенное свойство ордера. Свойство ордера должно быть типа datetime, int. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
long  HistoryOrderGetInteger(
   ulong                          ticket_number,     // тикет
   ENUM_ORDER_PROPERTY_INTEGER    property_id        // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  HistoryOrderGetInteger(
   ulong                          ticket_number,     // тикет
   ENUM_ORDER_PROPERTY_INTEGER    property_id,       // идентификатор свойства
   long&                          long_var           // сюда примем значение свойства
   );
```

Параметры

ticket_number

[in] Тикет ордера.

property_id

[in] Идентификатор свойства ордера. Значение может быть одним из значений перечисления ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] Переменная типа long, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа long.

Примечание

Не следует путать между собой ордера из торговой истории и действующие отложенные ордера, которые отображаются на вкладке "Торговля" в панели "Инструменты". Список ордеров, которые были отменены или привели к проведению торговой операции, можно посмотреть в закладке "История" на панели "Инструменты" клиентского терминала.

Пример:

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- получим тикет последнего ордера из истории торговли за неделю
   ulong last_order=GetLastOrderTicket();
```

© 2000-2025, MetaQuotes Ltd.

```
   if(HistoryOrderSelect(last_order))
     {
      //--- время постановки ордера в миллисекундах от 01.01.1970
      long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
      PrintFormat("Order #%d ORDER_TIME_SETUP_MSC=%i64 => %s",
                  last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
      //--- время исполнения/удаления ордера в миллисекундах от 01.01.1970
      long  time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
      PrintFormat("Order #%d ORDER_TIME_DONE_MSC=%i64 => %s",
                  last_order,time_done_msc,TimeToString(time_done_msc/1000));
     }
   else // сообщим о неудаче
      PrintFormat("HistoryOrderSelect() failed for #%d. Eror code=%d",
                  last_order,GetLastError());

//---
  }
//+------------------------------------------------------------------+
//| Возвращает тикет последнего ордера в истории или -1              |
//+------------------------------------------------------------------+
ulong GetLastOrderTicket()
  {
//--- запросим историю за последние 7 дней
   if(!GetTradeHistory(7))
     {
      //--- сообщим о неудачном вызове и вернем -1
      Print(__FUNCTION__," HistorySelect() вернул false");
      return -1;
     }
//---
   ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- если ордера есть, начинаем работать с ними
   if(orders>0)
     {
      Print("Orders = ",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;
     }
//--- не нашли ни одного ордера, вернем -1
   return -1;
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Запрашивает историю за последние дни и вернет false при неудаче  |
//+------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- зададим недельный период времени для запроса торговой истории
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- сделаем запрос и проверим результат
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- история получена успешно
   return true;
  }
```

Смотри также

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetString

Возвращает запрошенное свойство ордера. Свойство ордера должно быть типа string. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
string  HistoryOrderGetString(
   ulong                      ticket_number,     // тикет
   ENUM_ORDER_PROPERTY_STRING  property_id       // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  HistoryOrderGetString(
   ulong                      ticket_number,     // тикет
   ENUM_ORDER_PROPERTY_STRING  property_id,      // идентификатор свойства
   string&                    string_var         // сюда примем значение свойства
   );
```

Параметры

ticket_number

[in] Тикет ордера.

property_id

[in] Идентификатор свойства ордера. Значение может быть одним из значений перечисления ENUM_ORDER_PROPERTY_STRING.

string_var

[out] Переменная типа string, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа string.

Примечание

Не следует путать между собой ордера из торговой истории и действующие отложенные ордера, которые отображаются на вкладке "Торговля" в панели "Инструменты". Список ордеров, которые были отменены или привели к проведению торговой операции, можно посмотреть в закладке "История" на панели "Инструменты" клиентского терминала.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- запрашиваем историю сделок и ордеров
   if(!HistorySelect(0, TimeCurrent()))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("HistorySelect() failed. Error ", GetLastError());
       return;
      }

//--- в цикле по списку всех исторических ордеров на счёте
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет ордера в списке по индексу цикла
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип ордера и выводим заголовок для списка строковых свойств выбра
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket,
      PrintFormat("String properties of an history order %s #%I64u:", type, ticket);

      //--- распечатываем под заголовком все строковые свойства выбранного ордера
      HistoryOrderPropertiesStringPrint(ticket, 16);
     }
   /*
   результат:
   String properties of an history order Buy #2646074112:
   Comment:         [tp 1.09137]
   Symbol:          EURUSD
   External ID:
   String properties of an history order Buy #2646131906:
   Comment:
   Symbol:          EURUSD
   External ID:
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал строковые свойства                              |
//| выбранного исторического ордера                                  |
//+------------------------------------------------------------------+
void HistoryOrderPropertiesStringPrint(const long ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value="";

//--- определяем текст заголовка и ширину поля заголовка
//--- если ширина заголовка передана в функцию равной нулю, то шириной будет размер с
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- получаем и выводим в журнал комментарий с заголовком установленной ширины
   if(!HistoryOrderGetString(ticket, ORDER_COMMENT, value))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- выводим в журнал символ, по которому выставлен ордер
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetString(ticket, ORDER_SYMBOL, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- выводим в журнал идентификатор ордера во внешней системе
   header="External ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetString(ticket, ORDER_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа ордера                                  |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      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");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
```

Смотри также

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Свойства ордеров

© 2000-2025, MetaQuotes Ltd.

HistoryDealSelect

Выбирает в истории сделку для дальнейших обращений к ней через соответствующие функции. Возвращает true при успешном завершении функции. Возвращает false при неудачном завершении функции. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

```
bool  HistoryDealSelect(
   ulong  ticket     // тикет сделки
   );
```

Параметры

ticket

[in] Тикет сделки

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Не следует путать между собой ордера, сделки и позиции. Каждая сделка является результатом исполнения некоего ордера, каждая позиция является итоговым результатом одной или нескольких сделок.

Функция HistoryDealSelect() очищает в mql5-программе список сделок, доступных для обращений, и копирует в него одну единственную сделку, если выполнение HistoryDealSelect() завершилось успешно. Если необходимо перебрать все сделки, выбранные функцией HistorySelect(), то лучше использовать функцию HistoryDealGetTicket().

Пример:

```
#define   TICKET    2620919264   // тикет любой известной сделки, например, из истори

long      ExtTicket=TICKET;      // записываем заданный тикет в переменную из макропо
                                 // либо обрабатываем сделки в обработчике OnTradeTra
//+------------------------------------------------------------------+
//| Expert TradeTransaction handler                                  |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction& trans,
                        const MqlTradeRequest& request,
                        const MqlTradeResult& result)
  {
   //--- если транзакция - добавление сделки в историю
   if(trans.type==TRADE_TRANSACTION_DEAL_ADD)
     {
      //--- выбираем сделку по тикету, получаем её данные и выводим описание сделки в
      HistoryDealSelectProcess(trans.deal);
     }
  }

//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- выбираем сделку по тикету, получаем её данные и выводим описание сделки в журнал
   HistoryDealSelectProcess(ExtTicket);
   /*
   результат:
   (Position ID #2645974677) EURUSD Deal Out 0.10 Buy #2620919264 by order #264602896
   */
  }
//+------------------------------------------------------------------+
//| Выбирает сделку по тикету и распечатывает в журнале данные сделки|
//+------------------------------------------------------------------+
void HistoryDealSelectProcess(const ulong deal_ticket)
  {
//--- выбираем историческую сделку по тикету, указанному в deal_ticket
   ResetLastError();
   if(!HistoryDealSelect(deal_ticket))
     {
      PrintFormat("HistoryDealSelect(%I64u) failed. Error %d", deal_ticket, GetLastErr
      return;
     }

//--- если сделка успешно выбрана, получаем её данные и выводим описание сделки в журн
   ENUM_DEAL_TYPE    deal_type   = (ENUM_DEAL_TYPE)HistoryDealGetInteger(ExtTicket, DEA
   ENUM_DEAL_ENTRY   deal_entry  = (ENUM_DEAL_ENTRY)HistoryDealGetInteger(ExtTicket, DE
   ENUM_DEAL_REASON  deal_reason= (ENUM_DEAL_REASON)HistoryDealGetInteger(ExtTicket, D
   long              deal_time   = HistoryDealGetInteger(ExtTicket, DEAL_TIME_MSC);
   long              deal_order  = HistoryDealGetInteger(ExtTicket, DEAL_ORDER);
   long              deal_pos_id= HistoryDealGetInteger(ExtTicket, DEAL_POSITION_ID);
   string            deal_symbol= HistoryDealGetString(ExtTicket, DEAL_SYMBOL);
   double            deal_volume= HistoryDealGetDouble(ExtTicket, DEAL_VOLUME);
   double            deal_price  = HistoryDealGetDouble(ExtTicket, DEAL_PRICE);
   int               digits      = (int)SymbolInfoInteger(deal_symbol, SYMBOL_DIGITS);

   PrintFormat("(Position ID #%I64d) %s Deal %s %.2f %s #%I64u by order #%I64d at %.*
               deal_pos_id, deal_symbol, DealEntryDescription(deal_entry), deal_volume
               DealTypeDescription(deal_type), ExtTicket, deal_order, digits, deal_pri
               TimeMscToString(deal_time));
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа сделки                                  |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                    :  return("Buy");
```

© 2000-2025, MetaQuotes Ltd.

```
      case DEAL_TYPE_SELL                    :  return("Sell");
      case DEAL_TYPE_BALANCE                 :  return("Balance");
      case DEAL_TYPE_CREDIT                  :  return("Credit");
      case DEAL_TYPE_CHARGE                  :  return("Additional charge");
      case DEAL_TYPE_CORRECTION              :  return("Correction");
      case DEAL_TYPE_BONUS                   :  return("Bonus");
      case DEAL_TYPE_COMMISSION              :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY        :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY      :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY  :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY:  return("Monthly agent commission");
      case DEAL_TYPE_INTEREST                :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED            :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED           :  return("Canceled sell deal");
      case DEAL_DIVIDEND                     :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED             :  return("Franked (non-taxable) dividenc
      case DEAL_TAX                          :  return("Tax charges");
      default                                :  return("Unknown deal type: "+(string)t
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание способа изменения позиции                    |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("Reverce");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
//+------------------------------------------------------------------+
//| Возвращает время с миллисекундами                                |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
  }
```

Смотри также

HistorySelect(), HistoryDealGetTicket(), Свойства сделок

© 2000-2025, MetaQuotes Ltd.

HistoryDealsTotal

Возвращает количество сделок в истории. Перед вызовом функции HistoryDealsTotal() необходимо получить историю сделок и ордеров с помощью функции HistorySelect() или HistorySelectByPosition().

```
int  HistoryDealsTotal();
```

Возвращаемое значение

Значение типа int.

Примечание

Не следует путать между собой ордера, сделки и позиции. Каждая сделка является результатом исполнения некоего ордера, каждая позиция является итоговым результатом одной или нескольких сделок.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- запрашиваем всю имеющуюся историю на счёте
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- получаем количество сделок в списке и выводим значение в журнал
   int total=HistoryDealsTotal();
   Print("Number of historical deals on the account: ", total);
   /*
   результат:
   Number of historical deals on the account: 339
   */
  }
```

Смотри также

HistorySelect(), HistoryDealGetTicket(), Свойства сделок

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetTicket

Выбирает сделку для дальнейшей обработки и возвращает тикет сделки в истории. Перед вызовом функции HistoryDealGetTicket() необходимо получить историю сделок и ордеров с помощью функции HistorySelect() или HistorySelectByPosition().

```
ulong  HistoryDealGetTicket(
   int  index      // номер сделки
   );
```

Параметры

index

[in] Номер сделки в списке сделок.

Возвращаемое значение

Значение типа ulong. В случае неудачного выполнения возвращает 0.

Примечание

Не следует путать между собой ордера, сделки и позиции. Каждая сделка является результатом исполнения некоего ордера, каждая позиция является итоговым результатом одной или нескольких сделок.

Пример:

```
void OnStart()
  {
   ulong deal_ticket;             // тикет сделки
   ulong order_ticket;            // тикет ордера,по которому была совершена сделка
   datetime transaction_time;     // время совершения сделки
   long deal_type ;               // тип торговой операции
   long position_ID;              // идентификатор позиции
   string deal_description;       // описание операции
   double volume;                 // объем операции
   string symbol;                 // по какому символу была сделка
//--- установим начальную и конечную дату для запроса истории сделок
   datetime from_date=0;           // с самого начала
   datetime to_date=TimeCurrent();// по текущий момент
//--- запросим историю сделок в указанном интервале
   HistorySelect(from_date,to_date);
//--- общее количество в списке сделок
   int deals=HistoryDealsTotal();
//--- теперь обработаем каждую сделку
   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);
```

© 2000-2025, MetaQuotes Ltd.

```
      symbol=                    HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
      position_ID=               HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
      deal_description=          GetDealDescription(deal_type,volume,symbol,order_tic
      //--- сделаем красивое форматирование для номера сделки
      string print_index=StringFormat("% 3d",i);
      //--- выведем информацию по сделке
      Print(print_index+": deal #",deal_ticket," at ",transaction_time,deal_descriptio
     }
  }
//+------------------------------------------------------------------+
//| Возвращает строковое описание операции                           |
//+------------------------------------------------------------------+
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 ("correction");
      case DEAL_TYPE_BUY:                      descr="buy"; break;
      case DEAL_TYPE_SELL:                     descr="sell"; break;
      case DEAL_TYPE_BONUS:                    return ("bonus");
      case DEAL_TYPE_COMMISSION:               return ("additional commission");
      case DEAL_TYPE_COMMISSION_DAILY:         return ("daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY:       return ("monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY:   return ("daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return ("monthly agent commission");
      case DEAL_TYPE_INTEREST:                 return ("interest rate");
      case DEAL_TYPE_BUY_CANCELED:             descr="cancelled buy deal"; break;
      case DEAL_TYPE_SELL_CANCELED:            descr="cancelled sell deal"; break;
     }
   descr=StringFormat("%s %G %s (order #%d, position ID %d)",
                      descr,  // текущее описание
                      volume, // объем сделки
                      symbol, // инструмент сделки
                      ticket, // тикет ордера,вызвавшего сделку
                      pos_ID  // ID позиции, в которой участвовала сделка
                      );
   return(descr);
//---
  }
```

Смотри также

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), Свойства сделок

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetDouble

Возвращает запрошенное свойство сделки. Свойство сделки должно быть типа double. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
double  HistoryDealGetDouble(
   ulong                      ticket_number,     // тикет
   ENUM_DEAL_PROPERTY_DOUBLE  property_id        // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  HistoryDealGetDouble(
   ulong                      ticket_number,     // тикет
   ENUM_DEAL_PROPERTY_DOUBLE  property_id,       // идентификатор свойства
   double&                    double_var         // сюда примем значение свойства
   );
```

Параметры

ticket_number

[in] Тикет сделки.

property_id

[in] Идентификатор свойства сделки. Значение может быть одним из значений перечисления ENUM_DEAL_PROPERTY_DOUBLE.

double_var

[out] Переменная типа double, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа double.

Примечание

Не следует путать между собой ордера, сделки и позиции. Каждая сделка является результатом исполнения некоего ордера, каждая позиция является итоговым результатом одной или нескольких сделок.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- запрашиваем историю сделок и ордеров
   if(!HistorySelect(0, TimeCurrent()))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("HistorySelect() failed. Error ", GetLastError());
       return;
      }

//--- в цикле по списку сделок в истории счёта
   int total=HistoryDealsTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет очередной сделки (сделка автоматически выбирается для получ
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип и направление сделки и выводим заголовок для списка веществен
      string type=DealTypeDescription((ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket, DE
      string entry=DealEntryDescription((ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket,
      PrintFormat("Double properties of an deal %s entry %s #%I64u:", type, entry, tic

      //--- распечатываем под заголовком все вещественные свойства выбранной сделки
      HistoryDealPropertiesDoublePrint(ticket, 12);
     }
   /*
   результат:
   Double properties of an deal Buy entry In #2785070622:
   Volume:      0.50
   Price:       1.10480
   Commission: 0.00
   Swap:        0.00
   Profit:      0.00 USD
   Fee:         0.00
   StopLoss:    0.00000
   TakeProfit: 0.00000
   Double properties of an deal Sell entry Out #2785071538:
   Volume:      0.50
   Price:       1.10491
   Commission: 0.00
   Swap:        0.00
   Profit:      5.50 USD
   Fee:         0.00
   StopLoss:    0.00000
   TakeProfit: 0.00000
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал вещественные свойства  выбранной сделки         |
//+------------------------------------------------------------------+
void HistoryDealPropertiesDoublePrint(const ulong ticket, const uint header_width=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   uint    w=0;
   string header="";
   double value=0;

//--- получаем символ сделки, валюту прибыли и количество знаков после запятой для сим
   string symbol   = HistoryDealGetString(ticket, DEAL_SYMBOL);
   string currency= SymbolInfoString(symbol, SYMBOL_CURRENCY_PROFIT);
   int    digits   = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- определяем текст заголовка и ширину поля заголовка
//--- если ширина заголовка передана в функцию равной нулю, то шириной будет размер ст
   header="Volume:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- получаем и выводим в журнал объём сделки с заголовком установленной ширины
   if(!HistoryDealGetDouble(ticket, DEAL_VOLUME, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- выводим в журнал цену сделки
   header="Price:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_PRICE, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- выводим в журнал комиссию по сделке
   header="Commission:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_COMMISSION, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- выводим в журнал накопленный своп при закрытии
   header="Swap:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_SWAP, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- выводим в журнал финансовый результат сделки
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_PROFIT, value))
      return;
   PrintFormat("%-*s%-.2f %s", w, header, value, currency);

//--- выводим в журнал оплату за проведение сделки
   header="Fee:";
   w=(header_width==0 ? header.Length()+1 : header_width);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!HistoryDealGetDouble(ticket, DEAL_FEE, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- выводим в журнал значение уровня StopLoss
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- выводим в журнал значение уровня TakeProfit
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа сделки                                  |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                    :  return("Buy");
      case DEAL_TYPE_SELL                   :  return("Sell");
      case DEAL_TYPE_BALANCE                :  return("Balance");
      case DEAL_TYPE_CREDIT                 :  return("Credit");
      case DEAL_TYPE_CHARGE                 :  return("Additional charge");
      case DEAL_TYPE_CORRECTION             :  return("Correction");
      case DEAL_TYPE_BONUS                  :  return("Bonus");
      case DEAL_TYPE_COMMISSION             :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY       :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY     :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY:  return("Monthly agent commission");
      case DEAL_TYPE_INTEREST               :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED           :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED          :  return("Canceled sell deal");
      case DEAL_DIVIDEND                    :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED            :  return("Franked (non-taxable) dividend
      case DEAL_TAX                         :  return("Tax charges");
      default                               :  return("Unknown deal type: "+(string)t
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание способа изменения позиции                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("Reverce");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

Смотри также

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Свойства сделок

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetInteger

Возвращает запрошенное свойство сделки. Свойство сделки должно быть типа datetime, int. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
long  HistoryDealGetInteger(
   ulong                      ticket_number,     // тикет
   ENUM_DEAL_PROPERTY_INTEGER  property_id       // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  HistoryDealGetInteger(
   ulong                      ticket_number,     // тикет
   ENUM_DEAL_PROPERTY_INTEGER  property_id,       // идентификатор свойства
   long&                      long_var           // сюда примем значение свойства
   );
```

Параметры

ticket_number

[in] Тикет сделки.

property_id

[in] Идентификатор свойства сделки. Значение может быть одним из значений перечисления ENUM_DEAL_PROPERTY_INTEGER.

long_var

[out] Переменная типа long, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа long.

Примечание

Не следует путать между собой ордера, сделки и позиции. Каждая сделка является результатом исполнения некоего ордера, каждая позиция является итоговым результатом одной или нескольких сделок.

Пример:

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- получим тикет последней сделки из истории торговли за неделю
   ulong last_deal=GetLastDealTicket();
```

© 2000-2025, MetaQuotes Ltd.

```
   if(HistoryDealSelect(last_deal))
     {
      //--- время совершения сделки в миллисекундах от 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() failed for #%d. Eror code=%d",
                  last_deal,GetLastError());
//---
  }
//+------------------------------------------------------------------+
//| Возвращает тикет последней сделки в истории или -1               |
//+------------------------------------------------------------------+
ulong GetLastDealTicket()
  {
//--- запросим историю за последние 7 дней
   if(!GetTradeHistory(7))
     {
      //--- сообщим о неудачном вызове и вернем -1
      Print(__FUNCTION__," HistorySelect() вернул false");
      return -1;
     }
//---
   ulong first_deal,last_deal,deals=HistoryDealsTotal();
//--- если ордера есть, начинаем работать с ними
   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;
     }
//--- не нашли ни одной сделки, вернем -1
   return -1;
  }
//+------------------------------------------------------------------+
//| Запрашивает историю за последние дни и вернет false при неудаче  |
//+------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- зададим недельный период времени для запроса торговой истории
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- сделаем запрос и проверим результат
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- история получена успешно
   return true;
  }
```

Смотри также

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Свойства сделок

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetString

Возвращает запрошенное свойство сделки. Свойство сделки должно быть типа string. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
string  HistoryDealGetString(
   ulong                      ticket_number,     // тикет
   ENUM_DEAL_PROPERTY_STRING  property_id        // идентификатор свойства
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  HistoryDealGetString(
   ulong                      ticket_number,     // тикет
   ENUM_DEAL_PROPERTY_STRING  property_id,       // идентификатор свойства
   string&                    string_var         // сюда примем значение свойства
   );
```

Параметры

ticket_number

[in] Тикет сделки.

property_id

[in] Идентификатор свойства сделки. Значение может быть одним из значений перечисления ENUM_DEAL_PROPERTY_STRING.

string_var

[out] Переменная типа string, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа string.

Примечание

Не следует путать между собой ордера, сделки и позиции. Каждая сделка является результатом исполнения некоего ордера, каждая позиция является итоговым результатом одной или нескольких сделок.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- запрашиваем историю сделок и ордеров
   if(!HistorySelect(0, TimeCurrent()))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("HistorySelect() failed. Error ", GetLastError());
       return;
      }

//--- в цикле по списку сделок в истории счёта
   int total=HistoryDealsTotal();
   for(int i=0; i<total; i++)
     {
      //--- получаем тикет очередной сделки (сделка автоматически выбирается для получ
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;

      //--- получаем тип и направление сделки и выводим заголовок для списка веществен
      string type=DealTypeDescription((ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket, DE
      string entry=DealEntryDescription((ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket,
      PrintFormat("String properties of an deal %s entry %s #%I64u:", type, entry, tic

      //--- распечатываем под заголовком все вещественные свойства выбранной сделки
      HistoryDealPropertiesStringPrint(ticket, 13);
     }
   /*
   результат:
   String properties of an deal Buy entry In #2785021084:
   Symbol:      EURUSD
   Comment:     Test PositionGetString
   Extarnal ID:
   String properties of an deal Buy entry Out #2497993663:
   Symbol:      EURUSD
   Comment:     [tp 1.08639]
   Extarnal ID:
   */
  }
//+------------------------------------------------------------------+
//| Выводит в журнал строковые свойства  выбранной сделки            |
//+------------------------------------------------------------------+
void HistoryDealPropertiesStringPrint(const ulong ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value ="";

//--- определяем текст заголовка и ширину поля заголовка
//--- если ширина заголовка передана в функцию равной нулю, то шириной будет размер ст
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- получаем и выводим в журнал символ сделки с заголовком установленной ширины
   if(!HistoryDealGetString(ticket, DEAL_SYMBOL, value))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- выводим в журнал комментарий к сделке
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetString(ticket, DEAL_COMMENT, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- выводим в журнал идентификатор сделки во внешней торговой системе
   header="Extarnal ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetString(ticket, DEAL_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
//+------------------------------------------------------------------+
//| Возвращает описание типа сделки                                  |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                    :  return("Buy");
      case DEAL_TYPE_SELL                   :  return("Sell");
      case DEAL_TYPE_BALANCE                :  return("Balance");
      case DEAL_TYPE_CREDIT                 :  return("Credit");
      case DEAL_TYPE_CHARGE                 :  return("Additional charge");
      case DEAL_TYPE_CORRECTION             :  return("Correction");
      case DEAL_TYPE_BONUS                  :  return("Bonus");
      case DEAL_TYPE_COMMISSION             :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY       :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY     :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return("Monthly agent commission");
      case DEAL_TYPE_INTEREST               :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED           :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED          :  return("Canceled sell deal");
      case DEAL_DIVIDEND                    :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED            :  return("Franked (non-taxable) dividend
      case DEAL_TAX                         :  return("Tax charges");
      default                               :  return("Unknown deal type: "+(string)t
     }
  }
//+------------------------------------------------------------------+
//| Возвращает описание способа изменения позиции                    |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
```

© 2000-2025, MetaQuotes Ltd.

```
{
 switch(entry)
   {
    case DEAL_ENTRY_IN       :  return("In");
    case DEAL_ENTRY_OUT      :  return("Out");
    case DEAL_ENTRY_INOUT    :  return("Reverce");
    case DEAL_ENTRY_OUT_BY   :  return("Out by");
    case DEAL_ENTRY_STATE    :  return("Status record");
    default                  :  return("Unknown deal entry: "+(string)entry);
   }
}
```

Смотри также

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Свойства сделок

© 2000-2025, MetaQuotes Ltd.

Управление сигналами

Группа функций, предназначенных для управления торговыми сигналами. Данные функции позволяют:

- получать информацию о торговых сигналах, доступных для копирования,
- прочитать или установить настройки копирования торговых сигналов
- произвести подписку на сигнал и ее отмену средствами языка MQL5.

Функция	Действие
SignalBaseGetDouble	Возвращает значение свойства типа double для выбранного сигнала
SignalBaseGetInteger	Возвращает значение свойства типа integer для выбранного сигнала
SignalBaseGetString	Возвращает значение свойства типа string для выбранного сигнала
SignalBaseSelect	Выбирает для работы сигнал из базы торговых сигналов, доступных в терминале
SignalBaseTotal	Возвращает общее количество сигналов, доступных в терминале
SignalInfoGetDouble	Возвращает из настроек копирования торгового сигнала значение свойства типа double
SignalInfoGetInteger	Возвращает из настроек копирования торгового сигнала значение свойства типа integer
SignalInfoGetString	Возвращает из настроек копирования торгового сигнала значение свойства типа string
SignalInfoSetDouble	Устанавливает в настройках копирования торгового сигнала значение свойства типа double
SignalInfoSetInteger	Устанавливает в настройках копирования торгового сигнала значение свойства типа integer
SignalSubscribe	Производит подписку на копирование торгового сигнала
SignalUnsubscribe	Отменяет подписку на копирование торгового сигнала

© 2000-2025, MetaQuotes Ltd.

SignalBaseGetDouble

Возвращает значение свойства типа double для выбранного сигнала.

```
double  SignalBaseGetDouble(
   ENUM_SIGNAL_BASE_DOUBLE     property_id     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_BASE_DOUBLE.

Возвращаемое значение

Значение типа double указанного свойства сигнала.

© 2000-2025, MetaQuotes Ltd.

SignalBaseGetInteger

Возвращает значение свойства типа integer для выбранного сигнала.

```
long  SignalBaseGetInteger(
   ENUM_SIGNAL_BASE_INTEGER     property_id     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_BASE_INTEGER.

Возвращаемое значение

Значение типа integer указанного свойства сигнала.

© 2000-2025, MetaQuotes Ltd.

SignalBaseGetString

Возвращает значение свойства типа string для выбранного сигнала.

```
string  SignalBaseGetString(
   ENUM_SIGNAL_BASE_STRING     property_id     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_BASE_STRING.

Возвращаемое значение

Значение типа string указанного свойства сигнала.

© 2000-2025, MetaQuotes Ltd.

SignalBaseSelect

Выбирает для работы сигнал из базы торговых сигналов, доступных в терминале.

```
bool  SignalBaseSelect(
   int      index     // индекс записи сигнала
   );
```

Параметры

index

[in] Индекс записи сигнала в базе торговых сигналов.

Возвращаемое значение

Возвращает true при успешном завершении функции или false в случае ошибки. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
void OnStart()
  {
//--- запрашиваем общее количество сигналов в базе
   int total=SignalBaseTotal();
//--- цикл по всем сигналам
   for(int i=0;i<total;i++)
     {
      //--- выбираем сигнал для дальнейшей работы
      if(SignalBaseSelect(i))
        {
         //--- получение свойств сигнала
         long   id    =SignalBaseGetInteger(SIGNAL_BASE_ID);           // id сигнала
         long   pips  =SignalBaseGetInteger(SIGNAL_BASE_PIPS);         // результат тор
         long   subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS); // количество по
         string name  =SignalBaseGetString(SIGNAL_BASE_NAME);          // имя сигнала
         double price =SignalBaseGetDouble(SIGNAL_BASE_PRICE);         // цена подписки
         string curr  =SignalBaseGetString(SIGNAL_BASE_CURRENCY);      // валюта сигнал
         //--- выводим все прибыльные бесплатные сигналы с ненулевым количеством подпи
         if(price==0.0 && pips>0 && subscr>0)
            PrintFormat("id=%d, name=\"%s\", currency=%s, pips=%d, subscribers=%d",id,
        }
      else PrintFormat("Ошибка выбора сигнала. Код ошибки=%d",GetLastError());
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SignalBaseTotal

Возвращает общее количество сигналов, доступных в терминале.

```
int  SignalBaseTotal();
```

Возвращаемое значение

Общее количество сигналов, доступных в терминале.

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetDouble

Возвращает из настроек копирования торгового сигнала значение свойства типа double.

```
double  SignalInfoGetDouble(
   ENUM_SIGNAL_INFO_DOUBLE     property_id     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_INFO_DOUBLE.

Возвращаемое значение

Значение типа double указанного свойства настроек копирования торгового сигнала.

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetInteger

Возвращает из настроек копирования торгового сигнала значение свойства типа integer.

```
long  SignalInfoGetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_INFO_INTEGER.

Возвращаемое значение

Значение типа integer указанного свойства настроек копирования торгового сигнала.

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetString

Возвращает из настроек копирования торгового сигнала значение свойства типа string.

```
string  SignalInfoGetString(
   ENUM_SIGNAL_INFO_STRING     property_id     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Значение может быть одним из значений перечисления ENUM_SIGNAL_INFO_STRING.

Возвращаемое значение

Значение типа string указанного свойства настроек копирования торгового сигнала.

© 2000-2025, MetaQuotes Ltd.

SignalInfoSetDouble

Устанавливает в настройках копирования торгового сигнала значение свойства типа double.

```
bool  SignalInfoSetDouble(
   ENUM_SIGNAL_INFO_DOUBLE      property_id,     // идентификатор свойства
   double                       value            // значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_INFO_DOUBLE.

value

[in] Значение свойства настроек копирования торгового сигнала.

Возвращаемое значение

Возвращает true в случае успешного изменения свойства, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

SignalInfoSetInteger

Устанавливает в настройках копирования торгового сигнала значение свойства типа integer.

```
bool  SignalInfoSetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // идентификатор свойства
   long                         value            // значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_INFO_INTEGER.

value

[in] Значение свойства настроек копирования торгового сигнала.

Возвращаемое значение

Возвращает true в случае успешного изменения свойства, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

SignalSubscribe

Производит подписку на копирование указанного торгового сигнала.

```
bool  SignalSubscribe(
   long     signal_id     // id сигнала
   );
```

Параметры

signal_id

[in] Идентификатор сигнала.

Возвращаемое значение

Возвращает true в случае успешной подписки на копирование торгового сигнала, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

SignalUnsubscribe

Отменяет подписку на копирование торгового сигнала.

```
bool  SignalUnsubscribe();
```

Возвращаемое значение

Возвращает true в случае успешной отмены подписки на копирование торгового сигнала, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

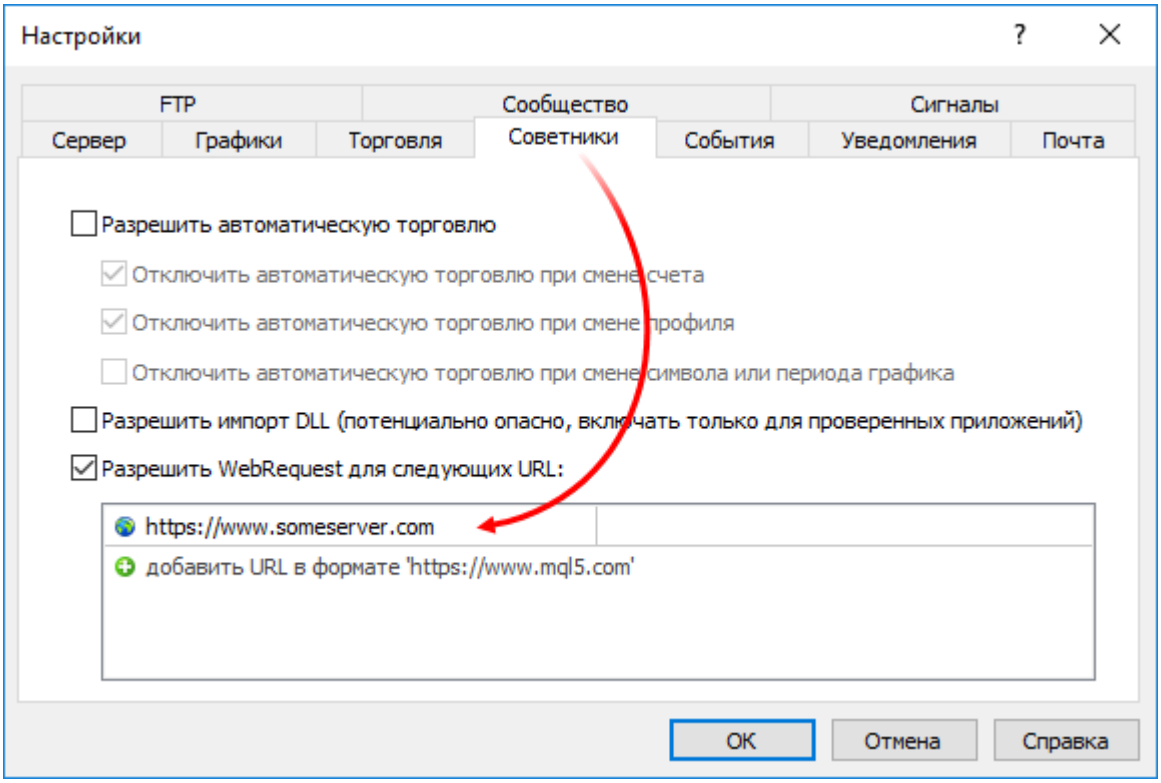
© 2000-2025, MetaQuotes Ltd.

Сетевые функции

Программы MQL5 могут обмениваться данными с удаленными серверами, отправлять push-уведомления, электронные письма и данные по FTP.

- Группа функций Socket* позволяет создать TCP-соединение (в том числе защищенное соединение TLS) с удаленным хостом через системные сокеты. Схема работы проста: вы создаете сокет, подключаетесь к серверу и можете начинать чтение и запись данных.
- Функция WebRequest предназначена для работы с веб-ресурсами и позволяет легко отправлять HTTP-запросы (в том числе GET и POST).
- SendFTP, SendMail и SendNotification — это более простые функции для отправки файлов, электронной почты и мобильных уведомлений.

Для безопасности конечного пользователя на стороне клиентского терминала реализован список разрешенных IP-адресов, с которыми может соединяться MQL5-программа при помощи функций Socket* и WebRequest. Например, если ей требуется подключение к https://www.someserver.com, то этот адрес должен быть явно указан пользователем терминала в списке разрешенных. Программно добавить адрес нельзя.



Чтобы уведомить пользователя о необходимости дополнительной настройки, добавьте в MQL5-программу явное сообщение. Например, через #property description, Alert или Print.

Функция	Действие
SocketCreate	Создает сокет с указанными флагами и возвращает его хэндл
SocketClose	Закрывает сокет
SocketConnect	Выполняет подключение к серверу с контролем таймаута
SocketIsConnected	Проверяет, подключен ли сокет в текущий момент времени

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
SocketIsReadable	Получает количество байт, которое можно прочитать из сокета
SocketIsWritable	Проверяет, возможна ли запись данных в сокет в текущий момент времени
SocketTimeouts	Устанавливает таймауты получения и отправки данных для системного объекта сокета
SocketRead	Читает данные из сокета
SocketSend	Записывает данные в сокете
SocketTlsHandshake	Инициирует защищенное TLS (SSL)-соединение с указанным хостом по протоколу TLS Handshake
SocketTlsCertificate	Получает данные о сертификате, используемом для защиты сетевого соединения
SocketTlsRead	Читает данные из защищенного TLS-соединения
SocketTlsReadAvailable	Читает все доступные данные из защищенного TLS-соединения
SocketTlsSend	Отправляет данные через защищенное TLS-соединение
WebRequest	Отправляет HTTP-запрос на указанный сервер
SendFTP	Посылает файл по адресу, указанному в окне настроек на закладке "FTP"
SendMail	Посылает электронное письмо по адресу, указанному в окне настроек на закладке "Почта"
SendNotification	Посылает Push-уведомления в мобильные терминалы, чьи MetaQuotes ID указаны на закладке "Уведомления"

© 2000-2025, MetaQuotes Ltd.

SocketCreate

Создает сокет с указанными флагами и возвращает его хэндл.

```
int  SocketCreate(
   uint     flags          // флаги
   );
```

Параметры

flags

[in] Комбинация флагов, определяющая режим работы с сокетом. В данный момент поддерживается один флаг — SOCKET_DEFAULT.

Возвращаемое значение

В случае успешного создания сокета возвращает его хэндл, иначе INVALID_HANDLE.

Примечания

Для освобождения памяти компьютера от неиспользуемого сокета вызовите для него SocketClose.

Из одной MQL5-программы можно создать максимум 128 сокетов. При превышении лимита в _LastError записывается ошибка 5271 (ERR_NETSOCKET_TOO_MANY_OPENED).

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Для работы примера добавьте Address в список разрешенных в нас
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
```

© 2000-2025, MetaQuotes Ltd.

```
   if(len<0)
      return(false);
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- разные команды чтения в зависимости от того, защищенное соединение или
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Получен заголовок HTTP ответа:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Установлено подключение к ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Сертификат TLS:");
            Print("   Владелец:  ",subject);
            Print("   Издатель:  ",issuer);
            Print("   Номер:     ",serial);
            Print("   Отпечаток: ",thumbprint);
            Print("   Истечение: ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET-запрос отправлен");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Не удалось получить ответ, ошибка ",GetLastError());
           }
         else
            Print("Не удалось отправить GET-запрос, ошибка ",GetLastError());
        }
      else
        {
         Print("Подключение к ",Address,":",Port," не удалось, ошибка ",GetLastError()
        }
      //--- закроем сокет после использования
      SocketClose(socket);
     }
   else
      Print("Не удалось создать сокет, ошибка ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketClose

Закрывает сокет.

```
bool  SocketClose(
   const int  socket      // хэндл сокета
   );
```

Параметры

socket

[in] Хэндл сокета, который необходимо закрыть. Хэндл возвращается функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Если для сокета ранее было создано соединение через SocketConnect, оно будет разорвано.

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Для работы примера добавьте Address в список разрешенных в нас
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- разные команды чтения в зависимости от того, защищенное соединение или
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Получен заголовок HTTP ответа:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Установлено подключение к ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Сертификат TLS:");
            Print("   Владелец:  ",subject);
            Print("   Издатель:  ",issuer);
            Print("   Номер:     ",serial);
            Print("   Отпечаток: ",thumbprint);
            Print("   Истечение: ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET-запрос отправлен");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Не удалось получить ответ, ошибка ",GetLastError());
           }
         else
            Print("Не удалось отправить GET-запрос, ошибка ",GetLastError());
        }
      else
        {
         Print("Подключение к ",Address,":",Port," не удалось, ошибка ",GetLastError()
        }
      //--- закроем сокет после использования
      SocketClose(socket);
     }
   else
      Print("Не удалось создать сокет, ошибка ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketConnect

Выполняет подключение к серверу с контролем таймаута.

```
bool  SocketConnect(
   int           socket,              // сокет
   const string  server,              // адрес для подключения
   uint          port,                // порт для подключения
   uint          timeout_receive_ms   // таймаут подключения
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

server

[in] Доменное имя сервера, к которому необходимо подключиться, или его IP-адрес.

port

[in] Номер порта для подключения.

timeout_receive_ms

[in] Таймаут подключения в миллисекундах. Если в течение этого времени подключение не удается выполнить, попытки останавливаются.

Возвращаемое значение

В случае успешного подключения возвращает true, иначе false.

Примечание

Адрес для подключения должен быть добавлен в список разрешенных на стороне клиентского терминала (раздел Сервис \ Настройки \ Советники).

При ошибке соединения в _LastError записывается ошибка 5272 (ERR_NETSOCKET_CANNOT_CONNECT).

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Для работы примера добавьте Address в список разрешенных в нас
```

© 2000-2025, MetaQuotes Ltd.

```
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- разные команды чтения в зависимости от того, защищенное соединение или
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Получен заголовок HTTP ответа:");
```

© 2000-2025, 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();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Установлено подключение к ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Сертификат TLS:");
            Print("   Владелец:  ",subject);
            Print("   Издатель:  ",issuer);
            Print("   Номер:     ",serial);
            Print("   Отпечаток: ",thumbprint);
            Print("   Истечение: ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET-запрос отправлен");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Не удалось получить ответ, ошибка ",GetLastError());
           }
         else
            Print("Не удалось отправить GET-запрос, ошибка ",GetLastError());
        }
      else
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         Print("Подключение к ",Address,":",Port," не удалось, ошибка ",GetLastError()
        }
      //--- закроем сокет после использования
      SocketClose(socket);
     }
   else
      Print("Не удалось создать сокет, ошибка ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketIsConnected

Проверяет, подключен ли сокет в текущий момент времени.

```
bool  SocketIsConnected(
   const int  socket      // хэндл сокета
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate(). При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Возвращаемое значение

Возвращает true, если сокет подключен, иначе false.

Примечание

С помощью функции SocketIsConnected() можно проверить подключение сокета в данный момент времени.

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                              SocketIsConnected.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                           https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal setting
#property script_show_inputs

input string Address    ="www.mql5.com";
input int    Port       =80;
input bool   CloseSocket=true;
bool         ExtTLS     =false;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- создаём сокет и получаем его хэндл
   int socket=SocketCreate();
//--- проверим хэндл
```

© 2000-2025, MetaQuotes Ltd.

```
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         PrintFormat("Established connection to %s:%d",Address,Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:       ",subject);
            Print("   Issuer:      ",issuer);
            Print("   Number:      ",serial);
            Print("   Print:       ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request sent");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         PrintFormat("Connection to %s:%d failed, error %d",Address,Port,GetLastError
        }
      //--- при установленном флаге закроем сокет после использования
      if(CloseSocket)
         SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());

//--- проверим подключение к серверу
   bool connected=SocketIsConnected(socket);
//--- если подключение отсутствует - завершаем работу
   if(!connected)
     {
      Print("No connection to server");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если есть подключение к серверу
   else
     {
      Print("Connection to the server is available\nThe connection needs to be closed.
      //--- закроем сокет и проверим ещё раз состояние подключения
      SocketClose(socket);
      connected=SocketIsConnected(socket);
     }

//--- распечатаем в журнале текущее состояние подключения к серверу
   Print("Currently connected: ",(connected ? "opened" : "closed"));
   /*
   результат при CloseSocket = true:
   No connection to server
   Currently connected: closed

   результат при CloseSocket = false:
   Connection to the server is available
   The connection needs to be closed. Closing
   Currently connected: closed
   */
  }
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
//--- конвертируем строку в массив символов, терминирующий ноль отбрасываем
   char req[];
   int  len=StringToCharArray(request,req)-1;

   if(len<0)
      return(false);
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
   char   rsp[];
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
```

© 2000-2025, MetaQuotes Ltd.

```
    {
     uint len=SocketIsReadable(socket);

     if(len)
       {
        int rsp_len;
        //--- разные команды чтения в зависимости от того, защищенное соединение или
        if(ExtTLS)
           rsp_len=SocketTlsRead(socket,rsp,len);
        else
           rsp_len=SocketRead(socket,rsp,len,timeout_ms);
        //--- разберем ответ
        if(rsp_len>0)
          {
           result+=CharArrayToString(rsp,0,rsp_len);
           //--- распечатаем только заголовок ответа
           int header_end=StringFind(result,"\r\n\r\n");

           if(header_end>0)
             {
              Print("HTTP answer header received:");
              Print(StringSubstr(result,0,header_end));
              return(true);
             }
           //--- обновим время истечения таймаута чтения
           timeout_check=GetTickCount64()+timeout_ms;
          }
       }
    }
  while(GetTickCount64()<timeout_check && !IsStopped());

  return(false);
 }
```

Смотри также

SocketConnect, SocketIsWritable, SocketCreate, SocketClose

© 2000-2025, MetaQuotes Ltd.

SocketIsReadable

Получает количество байт, которое можно прочитать из сокета.

```
uint  SocketIsReadable(
   const int  socket      // хэндл сокета
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Возвращаемое значение

Количество байт, которые можно прочитать. В случае ошибки возвращает 0.

Примечание

Если при выполнении этой функции на системном сокете произойдет ошибка, соединение, установленное через SocketConnect, будет разорвано.

Перед вызовом SocketRead проверяйте, есть ли в сокете данные для чтения. В ином случае, при отсутствии данных функция SocketRead будет впустую ожидать данные в течение timeout_ms, задерживая исполнение программы.

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Для работы примера добавьте Address в список разрешенных в нас
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
```

© 2000-2025, MetaQuotes Ltd.

```
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- разные команды чтения в зависимости от того, защищенное соединение или
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Получен заголовок HTTP ответа:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Установлено подключение к ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Сертификат TLS:");
            Print("   Владелец:  ",subject);
            Print("   Издатель:  ",issuer);
            Print("   Номер:     ",serial);
            Print("   Отпечаток: ",thumbprint);
            Print("   Истечение: ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET-запрос отправлен");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Не удалось получить ответ, ошибка ",GetLastError());
           }
         else
            Print("Не удалось отправить GET-запрос, ошибка ",GetLastError());
        }
      else
        {
         Print("Подключение к ",Address,":",Port," не удалось, ошибка ",GetLastError()
        }
      //--- закроем сокет после использования
      SocketClose(socket);
     }
   else
      Print("Не удалось создать сокет, ошибка ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketIsWritable

Проверяет, возможна ли запись данных в сокет в текущий момент времени.

```
bool  SocketIsWritable(
   const int  socket      // хэндл сокета
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Возвращаемое значение

Возвращает true, если запись возможна, иначе false.

Примечание

Используя эту функцию, вы можете проверить, возможна ли запись данных в сокет прямо сейчас.

Если при выполнении этой функции на системном сокете произойдет ошибка, соединение, установленное через SocketConnect, будет разорвано.

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                SocketIsWritable.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal setting
#property script_show_inputs

input string Address    ="www.mql5.com";
input int    Port       =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- создаём сокет и получаем его хэндл
   int socket=SocketCreate();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         PrintFormat("Established connection to %s:%d",Address,Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:       ",subject);
            Print("   Issuer:      ",issuer);
            Print("   Number:      ",serial);
            Print("   Print:       ",thumbprint);
            Print("   Expiration:  ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request sent");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         PrintFormat("Connection to %s:%d failed, error %d",Address,Port,GetLastError
        }
      //--- распечатаем в журнале возможность записи данных в сокет в текущий момент в
      if(SocketIsWritable(socket))
         Print("At the current moment in time, writing data to the socket is possible"
      else
         Print("It is not possible to write data to the socket at the current time");
      //--- закроем сокет после использования
      if(SocketClose(socket))
         Print("Now the socket is closed");
     }
   else
      Print("Failed to create a socket, error ",GetLastError());
   /*
   результат:
```

© 2000-2025, MetaQuotes Ltd.

```
   At the current moment in time, writing data to the socket is possible
   Socket is closed now
   */
  }
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
//--- конвертируем строку в массив символов, терминирующий ноль отбрасываем
   char req[];
   int  len=StringToCharArray(request,req)-1;

   if(len<0)
      return(false);
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
   char   rsp[];
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
     {
      uint len=SocketIsReadable(socket);

      if(len)
        {
         int rsp_len;
         //--- разные команды чтения в зависимости от того, защищенное соединение или
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout_ms);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");
```

© 2000-2025, MetaQuotes Ltd.

```
            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
            //--- обновим время истечения таймаута чтения
            timeout_check=GetTickCount64()+timeout_ms;
           }
        }
     }
  while(GetTickCount64()<timeout_check && !IsStopped());

  return(false);
 }
```

© 2000-2025, MetaQuotes Ltd.

SocketTimeouts

Устанавливает таймауты получения и отправки данных для системного объекта сокета.

```
bool  SocketTimeouts(
   int           socket,               // сокет
   uint          timeout_send_ms,      // таймаут отправки данных
   uint          timeout_receive_ms    // таймаут получения данных
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

timeout_send_ms

[in] Таймаут отправки данных в миллисекундах.

timeout_receive_ms

[in] Таймаут получения данных в миллисекундах.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Не путайте таймауты системных объектов и таймауты, устанавливаемые при чтении данных через SocketRead. SocketTimeout устанавливает таймауты один раз для объетка сокета в операционной системе. Эти таймауты будут применяться ко всем функциям чтения и отправки данных через этот сокет. В SocketRead таймаут устанавливается для конкретной операции чтения данных.

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                SocketTimeouts.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   TIMEOUT_SEND     5000
#define   TIMEOUT_RECEIVE  5000
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- создаём сокет и получаем его хэндл
   int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- устанавливаем таймауты получения и отправки данных для системного объекта сокета
   if(SocketTimeouts(socket,TIMEOUT_SEND,TIMEOUT_RECEIVE))
      PrintFormat("timeouts were successfully set",socket,TIMEOUT_SEND,TIMEOUT_RECEIVE
   else
      PrintFormat("SocketTimeouts(%d, %I64d, %I64d) failed. Error %d",socket,TIMEOUT_S
//--- закроем сокет
   SocketClose(socket);
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketRead

Читает данные из сокета.

```
int  SocketRead(
   int           socket,              // сокет
   uchar&        buffer[],            // буфер для чтения данных из сокета
   uint          buffer_maxlen,       // количество байт, которые нужно прочитать
   uint          timeout_ms           // таймаут чтения
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Ссылка на массив типа uchar, в который будут прочитаны данные. Размер динамического массива увеличивается на количество прочитанных байт. Размер массива не может превышать INT_MAX (2147483647).

buffer_maxlen

[in] Количество байт, которые необходимо прочитать в массив *buffer[]*. Данные, которые не поместятся в массив, останутся в сокете. Их можно будет получить следующим вызовом SocketRead. Значение *buffer_maxlen* не может превышать INT_MAX (2147483647).

timeout_ms

[in] Таймаут чтения данных в миллисекундах. Если в течение этого времени не удается получить данные, попытки завершаются и функция возвращает -1.

Возвращаемое значение

В случае успеха возвращает количество прочитанных байт, в случае ошибки возвращает -1.

Примечание

Если при выполнении этой функции на системном сокете произойдет ошибка, соединение, установленное через SocketConnect, будет разорвано.

При ошибке чтения данных в _LastError записывается ошибка 5273 (ERR_NETSOCKET_IO_ERROR).

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
```

© 2000-2025, MetaQuotes Ltd.

```
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Для работы примера добавьте Address в список разрешенных в нас
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- разные команды чтения в зависимости от того, защищенное соединение или
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");
```

© 2000-2025, MetaQuotes Ltd.

```
            if(header_end>0)
              {
               Print("Получен заголовок HTTP ответа:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Установлено подключение к ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Сертификат TLS:");
            Print("   Владелец:  ",subject);
            Print("   Издатель:  ",issuer);
            Print("   Номер:     ",serial);
            Print("   Отпечаток: ",thumbprint);
            Print("   Истечение: ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET-запрос отправлен");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Не удалось получить ответ, ошибка ",GetLastError());
           }
         else
            Print("Не удалось отправить GET-запрос, ошибка ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       else
         {
          Print("Подключение к ",Address,":",Port," не удалось, ошибка ",GetLastError()
         }
       //--- закроем сокет после использования
       SocketClose(socket);
      }
   else
       Print("Не удалось создать сокет, ошибка ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Смотри также

SocketTimeouts, MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketSend

Записывает данные в сокете.

```
int  SocketSend(
   int           socket,              // сокет
   const uchar&  buffer[],            // буфер для данных
   uint          buffer_len           // размер буфера
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[in] Ссылка на массив типа uchar с данными, которые необходимо отправить в сокет.

buffer_len

[in] Размер массива buffer.

Возвращаемое значение

В случае успеха возвращает количество байт, записанных в сокет. В случае ошибки возвращает -1.

Примечание

Если при выполнении этой функции на системном сокете произойдет ошибка, соединение, установленное через SocketConnect, будет разорвано.

При ошибке записи данных в _LastError записывается ошибка 5273 (ERR_NETSOCKET_IO_ERROR).

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Для работы примера добавьте Address в список разрешенных в нас
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
```

© 2000-2025, MetaQuotes Ltd.

```
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- разные команды чтения в зависимости от того, защищенное соединение или
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Получен заголовок HTTP ответа:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      }
    while(GetTickCount()<timeout_check && !IsStopped());
    return(false);
   }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Установлено подключение к ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Сертификат TLS:");
            Print("   Владелец:  ",subject);
            Print("   Издатель:  ",issuer);
            Print("   Номер:     ",serial);
            Print("   Отпечаток: ",thumbprint);
            Print("   Истечение: ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET-запрос отправлен");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Не удалось получить ответ, ошибка ",GetLastError());
           }
         else
            Print("Не удалось отправить GET-запрос, ошибка ",GetLastError());
        }
      else
        {
         Print("Подключение к ",Address,":",Port," не удалось, ошибка ",GetLastError()
        }
      //--- закроем сокет после использования
      SocketClose(socket);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   else
      Print("Не удалось создать сокет, ошибка ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Смотри также

SocketTimeouts, MathSwap, StringToCharArray

© 2000-2025, MetaQuotes Ltd.

SocketTlsHandshake

Инициирует защищенное TLS (SSL)-соединение с указанным хостом по протоколу TLS Handshake. Во время Handshake клиент и сервер согласовывают параметры соединения: версию используемого протокола и способ шифрования данных.

```
bool  SocketTlsHandshake(
   int           socket,               // сокет
   const string  host                  // адрес хоста
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

host

[in] Адрес хоста, с которым устанавливается защищенное соединение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечания

До защищенного соединения программа должна установить обычное TCP-соединение с хостом при помощи SocketConnect.

При ошибке установления защищенного соединения в _LastError записывается ошибка 5274 (ERR_NETSOCKET_HANDSHAKE_FAILED).

Вызов этой функции не требуется, если подключение осуществляется к порту 443. Это стандартный TCP-порт, используемый для защищенных TLS (SSL)-подключений.

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                              SocketTlsHandshake.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   SERVER    "smtp.gmail.com"
#define   PORT      587
//+------------------------------------------------------------------+
//| Ручное переключение небезопасного соединения на безопасное       |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool TlsHandshake(int socket)
  {
//--- получим приветствие от сервера
   string rsp;

   if(!RecvString(socket,rsp))
      return(false);
//--- приветствуем сервер
   if(!SendString(socket,"EHLO my.domain.com\r\n"))
      return(false);
//--- получаем ответ сервера со списком поддерживаемых команд
   if(!RecvString(socket,rsp))
      return(false);
//--- распечатаем приветствие
   Print("SERVER: ",rsp);
//--- информируем сервер, что хотим перейти с небезопасного соединения на безопасное
   if(!SendString(socket,"STARTTLS\r\n"))
      return(false);
//--- получаем ответ сервера
   if(!RecvString(socket,rsp))
      return(false);
//--- в примере мы не проверяем ответ сервера о готовности перейти на TLS ('Ready to s

//--- инициируем защищенное TLS (SSL)-соединение с указанным хостом по протоколу TLS
   if(SocketTlsHandshake(socket,SERVER))
      return(true);

   Print("SocketTlsHandshake() failed. Error ",GetLastError());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- создаём сокет и получаем его хэндл
   int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- подключаемся к серверу SERVER по порту PORT
   if(!SocketConnect(socket,SERVER,PORT,10000))
     {
      Print("SocketConnect() failed. Error ",GetLastError());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- установлено небезопасное соединение
      PrintFormat("%s connection has been established to %s:%d",(PORT==443 ? "A secure
      //--- переключаем на безопасное соединение
      if(PORT!=443 && TlsHandshake(socket))
        {
         PrintFormat("Unsecured connection to %s:%d switched to secured",SERVER,PORT);
         //--- если соединение защищено сертификатом, выведем его данные
         string   subject,issuer,serial,thumbprint;
         datetime expiration;

         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:      ",subject);
            Print("   Issuer:     ",issuer);
            Print("   Number:     ",serial);
            Print("   Print:      ",thumbprint);
            Print("   Expiration: ",expiration);
           }
        }
     }
//--- закроем сокет после использования
   SocketClose(socket);
   /*
   результат:
   An unsecured connection has been established to smtp.gmail.com:587
   SERVER: 220 smtp.gmail.com ESMTP a640c23a62f3a-a9b1f298319sm82305866b.105 - gsmtp

   SERVER: 250-smtp.gmail.com at your service, [37.193.40.122]
   250-SIZE 35882577
   250-8BITMIME
   250-STARTTLS
   250-ENHANCEDSTATUSCODES
   250-PIPELINING
   250-CHUNKING
   250 SMTPUTF8

   SERVER: 220 2.0.0 Ready to start TLS

   SocketTlsHandshake(): A secure connection to smtp.gmail.com:587 is now established
   TLS certificate:
      Owner:  /CN=smtp.gmail.com
      Issuer:  /C=US/O=Google Trust Services/CN=WR2
      Number:     1f:f4:db:2a:5a:e6:dc:52:0a:4c:05:ce:81:cc:c3:f7
      Print: d6be8af229b5329cd3d4c2789c02aa94f89b421c
      Expiration: 2024.12.30 08:25:30
   */
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Отправляет строку на сервер                                      |
//+------------------------------------------------------------------+
bool SendString(int socket,const string str)
  {
//--- конвертируем строку в массив символов
   uchar data[];
   int   size=StringToCharArray(str,data,0,str.Length(),CP_UTF8);
//--- отправляем данные в сокет
   ResetLastError();
   if(SocketSend(socket,data,size)==size)
      return(true);
//--- ошибка отправки данных
   Print("Failed to send data to server. Error ",GetLastError());
   return false;
  }
//+------------------------------------------------------------------+
//| Получает строку с сервера                                        |
//+------------------------------------------------------------------+
bool RecvString(int socket,string& result,uint timeout_ms=1000)
  {
//--- ждём появления данных на сокете
   ulong wait_time_end=GetMicrosecondCount()+timeout_ms*1000;

   while(!SocketIsReadable(socket))
     {
      Sleep(10);
      //--- время ожидания данных истекло - возвращаем NULL в качестве ответа
      if(wait_time_end<GetMicrosecondCount())
        {
         Print("ERROR: No response from server");
         return(false);
        }
     }
//--- читаем данные из сокета
   uchar  data[128];
   uint   size=0;
   string resp=NULL;

   do
     {
      uchar b[1];
      int   n=SocketRead(socket,b,1,1000);

      if(n < 0)
         break;

      if(n)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          data[size++]=b[0];

          if(size==data.Size())
            {
             resp += CharArrayToString(data,0,data.Size(),CP_UTF8);
             size = 0;
            }
         }
      }
   while(SocketIsReadable(socket));
//--- копируем в строку прочитанные данные
   if(size)
      resp+=CharArrayToString(data,0,size,CP_UTF8);
//--- если пустая строка, то ошибка
   if(!resp.Length())
     {
      Print("ERROR: No response from server");
      return(false);
     }
//--- отдаём строку
   result=resp;
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketTlsCertificate

Получает данные о сертификате, используемом для защиты сетевого соединения.

```
bool  SocketTlsCertificate(
   int           socket,               // сокет
   string&       subject,              // владелец сертификата
   string&       issuer,               // издатель сертификата
   string&       serial,               // серийный номер сертификата
   string&       thumbprint,           // отпечаток сертификата
   datetime&     expiration            // срок истечения сертификата
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

subject

[in] Имя владельца сертификата. Соответствует полю Subject.

issuer

[in] Имя издателя сертификата. Соответствует полю Issuer.

serial

[in] Серийный номер сертификата. Соответствует полю SerialNumber.

thumbprint

[in] Отпечаток сертификата. Соответствует хэшу SHA-1 от всего файла сертификата (все поля, включая подпись издателя).

expiration

[in] Срок истечения сертификата в формате datetime.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Запрос данных от сертификате возможен только после установления защищенного соединения при помощи SocketTlsHandshake.

При ошибке получения сертификата в _LastError записывается ошибка 5275 (ERR_NETSOCKET_NO_CERTIFICATE).

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
```

© 2000-2025, 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 "Для работы примера добавьте Address в список разрешенных в нас
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- разные команды чтения в зависимости от того, защищенное соединение или
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- разберем ответ
```

© 2000-2025, MetaQuotes Ltd.

```
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Получен заголовок HTTP ответа:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Установлено подключение к ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Сертификат TLS:");
            Print("   Владелец:  ",subject);
            Print("   Издатель:  ",issuer);
            Print("   Номер:     ",serial);
            Print("   Отпечаток: ",thumbprint);
            Print("   Истечение: ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET-запрос отправлен");
            //--- прочитаем ответ
```

© 2000-2025, MetaQuotes Ltd.

```
            if(!HTTPRecv(socket,1000))
               Print("Не удалось получить ответ, ошибка ",GetLastError());
            }
         else
            Print("Не удалось отправить GET-запрос, ошибка ",GetLastError());
        }
      else
        {
         Print("Подключение к ",Address,":",Port," не удалось, ошибка ",GetLastError()
        }
      //--- закроем сокет после использования
      SocketClose(socket);
     }
   else
      Print("Не удалось создать сокет, ошибка ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketTlsRead

Читает данные из защищенного TLS-соединения.

```
int  SocketTlsRead(
   int           socket,              // сокет
   uchar&        buffer[],            // буфер для чтения данных из сокета
   uint          buffer_maxlen        // количество байт, которые нужно прочитать
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Ссылка на массив типа uchar, в который будут прочитаны данные. Размер динамического массива увеличивается на количество прочитанных байт. Размер массива не может превышать INT_MAX (2147483647).

buffer_maxlen

[in] Количество байт, которые необходимо прочитать в массив *buffer[]*. Данные, которые не поместятся в массив, останутся в сокете. Их можно будет получить следующим вызовом SocketTLSRead. Значение *buffer_maxlen* не может превышать INT_MAX (2147483647).

Возвращаемое значение

В случае успеха возвращает количество прочитанных байт, в случае ошибки возвращает -1.

Примечание

Если при выполнении этой функции на системном сокете произойдет ошибка, соединение, установленное через SocketConnect, будет разорвано.

Функция исполняется до тех пор, пока не получит указанное количество данных или не наступит таймаут (SocketTimeouts).

При ошибке чтения данных в _LastError записывается ошибка 5273 (ERR_NETSOCKET_IO_ERROR).

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                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"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Для работы примера добавьте Address в список разрешенных в нас
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Отправка команды на сервер                                       |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- если используется защищенное TLS-соединение через порт 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- если используется обычное TCP-соединение
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- разные команды чтения в зависимости от того, защищенное соединение или
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- разберем ответ
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
```

© 2000-2025, MetaQuotes Ltd.

```
               Print("Получен заголовок HTTP ответа:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Установлено подключение к ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Сертификат TLS:");
            Print("   Владелец:  ",subject);
            Print("   Издатель:  ",issuer);
            Print("   Номер:     ",serial);
            Print("   Отпечаток: ",thumbprint);
            Print("   Истечение: ",expiration);
            ExtTLS=true;
           }
         //--- отправим на сервер запрос GET
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET-запрос отправлен");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Не удалось получить ответ, ошибка ",GetLastError());
           }
         else
            Print("Не удалось отправить GET-запрос, ошибка ",GetLastError());
        }
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          Print("Подключение к ",Address,":",Port," не удалось, ошибка ",GetLastError()
         }
       //--- закроем сокет после использования
       SocketClose(socket);
      }
    else
       Print("Не удалось создать сокет, ошибка ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Смотри также

SocketTimeouts, MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketTlsReadAvailable

Читает все доступные данные из защищенного TLS-соединения.

```
int  SocketTlsReadAvailable(
   int           socket,              // сокет
   uchar&        buffer[],            // буфер для чтения данных из сокета
   const uint    buffer_maxlen        // количество байт, которые нужно прочитать
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Ссылка на массив типа uchar, в который будут прочитаны данные. Размер динамического массива увеличивается на количество прочитанных байт. Размер массива не может превышать INT_MAX (2147483647).

buffer_maxlen

[in] Количество байт, которые необходимо прочитать в массив buffer[]. Данные, которые не поместятся в массив, останутся в сокете. Их можно будет получить следующим вызовом SocketTlsReadAvailable или SocketTlsRead. Значение buffer_maxlen не может превышать INT_MAX (2147483647).

Возвращаемое значение

В случае успеха возвращает количество прочитанных байт, в случае ошибки возвращает -1.

Примечание

Если при выполнении этой функции на системном сокете произойдет ошибка, соединение, установленное через SocketConnect, будет разорвано.

При ошибке чтения данных в _LastError записывается ошибка 5273 (ERR_NETSOCKET_IO_ERROR).

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                      SocketTlsReadAvailable.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property script_show_inputs
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Входные параметры скрипта                                        |
//+------------------------------------------------------------------+
input string InpMethod ="GET";             // Method (HEAD,GET)
input string InpServer ="www.google.com"; // Server
input uint   InpPort   =443;               // Port
input uint   InpTimeout=5000;              // Timeouts
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
   Print("Server: ",InpServer);
   Print("Port: ",InpPort);
//--- создаём сокет и получаем его хэндл
   const int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- устанавливаем таймауты получения и отправки данных для системного объекта сокета
   if(!SocketTimeouts(socket,InpTimeout,InpTimeout))
     {
      PrintFormat("SocketTimeouts(%u, %u) failed. Error %d",InpTimeout,InpTimeout,GetL
      SocketClose(socket);
      return;
     }
//--- подключаемся к серверу Server по порту Port
   if(!SocketConnect(socket,InpServer,InpPort,InpTimeout))
     {
      PrintFormat("SocketConnect('%s', %u, %u) failed. Error %d",InpServer,InpPort,Inp
      SocketClose(socket);
      return;
     }
//--- получаем данные о сертификате, используемом для защиты сетевого соединения
   string   subject,issuer,serial,thumbprint;
   datetime expiration;

   if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
     {
      Print("TLS certificate:");
      Print("   Owner:      ",subject);
      Print("   Issuer:     ",issuer);
      Print("   Number:     ",serial);
      Print("   Print:      ",thumbprint);
      Print("   Expiration: ",expiration);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- сервер не предоставляет сертификат - сообщаем о незащищенном соединении и
      Print("The connection is not secured by a certificate");
      SocketClose(socket);
      return;
     }
//--- отправляем запрос на сервер
   string request=StringFormat("%s / HTTP/1.1\r\nHost: %s\r\nUser-Agent: MetaTrader 5\

   if(HTTPSendTLS(socket,request))
     {
      //--- запрос отправлен - получаем ответ
      Print("\nRequest sent. Starting page loading...");
      uchar response[]; // все полученные данные (заголовок и тело документа)

      if(!HTTPRecvTLS(socket,response,InpTimeout))
        {
         Print("There were errors while reading the page");
         SocketClose(socket);
         return;
        }
      //--- сообщим о количестве байт принятых данных
      PrintFormat("%u bytes received",response.Size());
      //--- распечатаем только заголовок полученной страницы
      string result    =CharArrayToString(response,0,WHOLE_ARRAY,CP_UTF8);
      int    header_end=StringFind(result,"\r\n\r\n");

      if(header_end>0)
        {
         Print("\nHTTP answer header received:");
         Print(StringSubstr(result,0,header_end));
        }
     }
//--- закроем сокет после использования
   SocketClose(socket);
   /*
   результат:
   Server: www.google.com
   Port: 443
   TLS certificate:
      Owner: /CN=www.google.com
      Issuer: /C=US/O=Google Trust Services/CN=WR2
      Number: 0d:43:b1:4a:bb:9c:15:96:10:e1:3d:55:23:9f:25:4e
      Print: 89167618e5017f813aff981c88ce422dc1016bdf
      Expiration: 2024.12.30 08:26:35

   Request sent. Starting page loading...
   HTTPRecvTLS: Document received within 27 attempts
```

© 2000-2025, MetaQuotes Ltd.

```
   25185 bytes received

   HTTP answer header received:
   HTTP/1.1 200 OK
   Date: Fri, 25 Oct 2024 17:12:42 GMT
   Expires: -1
   Cache-Control: private, max-age=0
   Content-Type: text/html; charset=ISO-8859-1
   Content-Security-Policy-Report-Only: object-src 'none';base-uri 'self';script-src '
   Accept-CH: Sec-CH-Prefers-Color-Scheme
   P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
   Server: gws
   X-XSS-Protection: 0
   X-Frame-Options: SAMEORIGIN
   Set-Cookie: AEC=AVYB7coyYMCdweTDTaWeGYzmRnxzKGqsOEosH_VkbCn8xhWkFz6v0kxQFw; expires
   Set-Cookie: NID=518=J02X02Ff4v_9sMcNoUz-1SolmuG08E26Gs438ik0J_SOJUMy7of-P-qup-LaNSW
   Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
   Accept-Ranges: none
   Vary: Accept-Encoding
   Transfer-Encoding: chunked
   */
  }
//+------------------------------------------------------------------+
//| Отправляет HTTP-запрос через защищенное  соединение              |
//+------------------------------------------------------------------+
bool HTTPSendTLS(int socket,const string request)
  {
//--- конвертируем строку в массив символов, терминирующий ноль отбрасываем
   char req[];
   int  len=StringToCharArray(request,req,0,WHOLE_ARRAY,CP_UTF8)-1;

   if(len<0)
      return false;

   return(SocketTlsSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Получает веб-страницу через безопасное соединение                |
//+------------------------------------------------------------------+
bool HTTPRecvTLS(int socket,uchar &response[],const uint timeout_ms)
  {
//--- читаем доступные данные из защищенного TLS-соединения, пока не истекло время ожи
   ulong timeout_check=GetTickCount64()+timeout_ms;
   uchar block[1024];   // буфер блочного чтения данных из сокета
   uint  attempt=0;     // запрошенное количество блоков данных
   int   err    =0;     // код ошибки

   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   do
     {
      //--- читаем блоками, максимум по 1024 байта
      int len=SocketTlsReadAvailable(socket,block,1024);

      if(len>0)
        {
         attempt++;
         //--- объединяем полученные блоки данных
         ArrayCopy(response,block,response.Size());
         //--- здесь анализируем полученные данные, определяем заголовок, тело страни
         //...
         //...
         //...
         timeout_check=GetTickCount64()+timeout_ms;
        }
      else
         Sleep(10);

      err=GetLastError();
     }
   while(!IsStopped() && GetTickCount()<timeout_check && !err);
//--- при чтении были ошибки?
   if(err)
     {
      Print("Error ",err);
      return(false);
     }

   PrintFormat("%s: Document received within %d attempts",__FUNCTION__,attempt);
   return(true);
  }
```

Смотри также

SocketTimeouts, MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketTlsSend

Отправляет данные через защищенное TLS-соединение.

```
int  SocketTlsSend(
   int           socket,              // сокет
   const uchar&  buffer[],            // буфер для данных
   uint          buffer_len           // размер буфера
   );
```

Параметры

socket

[in] Хэндл сокета, возвращаемый функцией SocketCreate. При передаче неверного хэндла в _LastError записывается ошибка 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[in] Ссылка на массив типа uchar с данными, которые необходимо отправить.

buffer_len

[in] Размер массива buffer.

Возвращаемое значение

В случае успеха возвращает количество байт, записанных в сокет. В случае ошибки возвращает -1.

Примечание

Если при выполнении этой функции на системном сокете произойдет ошибка, соединение, установленное через SocketConnect, будет разорвано.

При ошибке записи данных в _LastError записывается ошибка 5273 (ERR_NETSOCKET_IO_ERROR).

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

Пример:

```
//+------------------------------------------------------------------+
//|                                                SocketTlsSend.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "Add Address to the list of allowed ones in the terminal setting
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =443;
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
   int socket=SocketCreate();
//--- проверим хэндл
   if(socket!=INVALID_HANDLE)
     {
      //--- если всё в порядке, подключаемся
      if(SocketConnect(socket,Address,Port,1000))
        {
         PrintFormat("Established connection to %s:%d",Address,Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- если соединение защищено сертификатом, выведем его данные
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:      ",subject);
            Print("   Issuer:     ",issuer);
            Print("   Number:     ",serial);
            Print("   Print:      ",thumbprint);
            Print("   Expiration: ",expiration);
           }
         //--- отправим на сервер запрос GET
         string request="GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r\n\r
         char   req[];
         int    len=StringToCharArray(request,req)-1;

         if(len<0)
           {
            Print("StringToCharArray() failed. Error ", GetLastError());
            SocketClose(socket);
            return;
           }
         //--- если используется защищенное TLS-соединение через порт 443
         if(SocketTlsSend(socket,req,len)==len)
           {
            Print("GET request sent");
            //--- прочитаем ответ
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          PrintFormat("Connection to %s:%d failed, error %d",Address,Port,GetLastError
         }
       //--- закроем сокет после использования
       SocketClose(socket);
      }
    else
       Print("Failed to create a socket, error ",GetLastError());
  }
//+------------------------------------------------------------------+
//| Чтение ответа сервера                                            |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
//--- читаем данные из сокета, пока они есть, но не дольше timeout
   char   rsp[];
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;

   do
     {
      uint len=SocketIsReadable(socket);

      if(len)
        {
         //--- читаем и разбираем данные защищенного TLS-соединения
         int rsp_len=SocketTlsRead(socket,rsp,len);

         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- распечатаем только заголовок ответа
            int header_end=StringFind(result,"\r\n\r\n");

            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
            //--- обновим время истечения таймаута чтения
            timeout_check=GetTickCount64()+timeout_ms;
           }
        }
     }
   while(GetTickCount64()<timeout_check && !IsStopped());

   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

SocketTimeouts, MathSwap, StringToCharArray

© 2000-2025, MetaQuotes Ltd.

WebRequest

Отправляет HTTP-запрос на указанный сервер. Существует два варианта функции:

1. **Для отправки простых запросов** вида "ключ=значение" с использованием заголовка Content-Type: application/x-www-form-urlencoded.

```
int  WebRequest(
   const string      method,           // метод HTTP
   const string      url,              // url-адрес
   const string      cookie,           // cookie
   const string      referer,          // referer
   int               timeout,          // таймаут
   const char        &data[],          // массив тела HTTP-сообщения
   int               data_size,        // размер массива data[] в байтах
   char              &result[],        // массив с данными ответа сервера
   string            &result_headers   // заголовки ответа сервера
   );
```

2. **Для отправки запросов произвольного типа** с указанием собственного набора заголовков для более гибкого взаимодействия с различными Web-сервисами.

```
int  WebRequest(
   const string      method,           // метод HTTP
   const string      url,              // url-адрес
   const string      headers,          // заголовки
   int               timeout,          // таймаут
   const char        &data[],          // массив тела HTTP-сообщения
   char              &result[],        // массив с данными ответа сервера
   string            &result_headers   // заголовки ответа сервера
   );
```

Параметры

method

[in] Метод HTTP.

url

[in] URL-адрес.

headers

[in] Заголовки запроса вида "ключ: значение", разделенные переносом строки "\r\n".

cookie

[in] Значение Cookie.

referer

[in] Значение заголовка Referer HTTP-запроса.

timeout

[in] Таймаут в миллисекундах.

© 2000-2025, MetaQuotes Ltd.

data[]

[in] Массив данных тела HTTP-сообщения.

data_size

[in] Размер массива data[].

result[]

[out] Массив с данными ответа сервера.

result_headers

[out] Заголовки ответа сервера.

Возвращаемое значение

Код ответа HTTP-сервера либо -1 в случае ошибки.

Примечание

Для использования функции WebRequest() следует добавить адреса серверов в список разрешенных URL во вкладке "Советники" окна "Настройки". Порт сервера выбирается автоматически на основе указанного протокола - 80 для "http://" и 443 для "https://".

Функция WebRequest() является синхронной, это означает, что она приостанавливает выполнение программы и ждет ответа от запрашиваемого сервера. Так как задержки при получении ответа на отправленный запрос могут быть большими, то функция запрещена для вызовов из индикаторов, поскольку индикаторы работают в едином потоке, общем для всех индикаторов и графиков на данном символе. Задержка выполнения индикатора на одном из графиков символа может привести к остановке обновления всех графиков по данному символу.

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Системная функция не разрешена для вызова".

При работе в тестере стратегий функция WebRequest() не выполняется.

Пример

```
void OnStart()
  {
   string cookie=NULL,headers;
   char   post[],result[];
   string url="https://finance.yahoo.com";
//--- для работы с сервером необходимо добавить URL "https://finance.yahoo.com"
//--- в список разрешенных URL (Главное меню->Сервис->Настройки, вкладка "Советники");
//--- обнуляем код последней ошибки
   ResetLastError();
//--- загрузка html-страницы с Yahoo Finance
   int res=WebRequest("GET",url,cookie,NULL,500,post,0,result,headers);
   if(res==-1)
     {
      Print("Ошибка в WebRequest. Код ошибки  =",GetLastError());
      //--- возможно, URL отсутствует в списке, выводим сообщение о необходимости его
      MessageBox("Необходимо добавить адрес '"+url+"' в список разрешенных URL во вкла
```

© 2000-2025, MetaQuotes Ltd.

```
    }
  else
    {
     if(res==200)
       {
        //--- успешная загрузка
        PrintFormat("Файл успешно загружен, размер %d байт.",ArraySize(result));
        //PrintFormat("Заголовки сервера: %s",headers);
        //--- сохраняем данные в файл
        int filehandle=FileOpen("url.htm",FILE_WRITE|FILE_BIN);
        if(filehandle!=INVALID_HANDLE)
          {
           //--- сохраняем содержимое массива result[] в файл
           FileWriteArray(filehandle,result,0,ArraySize(result));
           //--- закрываем файл
           FileClose(filehandle);
          }
        else
           Print("Ошибка в FileOpen. Код ошибки =",GetLastError());
       }
     else
        PrintFormat("Ошибка загрузки '%s', код %d",url,res);
    }
}
```

© 2000-2025, MetaQuotes Ltd.

SendFTP

Посылает файл по адресу, указанному в окне настроек на закладке "FTP".

```
bool  SendFTP(
   string  filename,           // файл для отсылки по ftp
   string  ftp_path=NULL       // путь для выгрузки на ftp-сервере
   );
```

Параметры

filename

[in] Имя отсылаемого файла.

ftp_path=NULL

[in] Каталог FTP. Если каталог не указан, то используется каталог, описанный в настройках.

Возвращаемое значение

В случае неудачи возвращает false.

Примечание

Отсылаемый файл должен находиться в папке *каталог_терминала\MQL5\files* или ее подпапках. Отсылка не производится, если в настройках не указан адрес FTP и/или пароль доступа.

При работе в тестере стратегий функция SendFTP() не выполняется.

Пример:

```
//+------------------------------------------------------------------+
//|                                                      SendFTP.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   FILENAME  "SomeFile.bin"
#define   PATH      NULL
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- проверим разрешение в терминале на отправку файлов на FTP-адрес
   if(!TerminalInfoInteger(TERMINAL_FTP_ENABLED))
     {
      Print("Error. The client terminal does not have permission to send messages to t
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- отправляем файл
   ResetLastError();
   if(!SendFTP(FILENAME, PATH))
      Print("SendFTP() failed. Error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

SendMail

Посылает электронное письмо по адресу, указанному в окне настроек на закладке "Почта".

```
bool  SendMail(
   string  subject,        // заголовок
   string  some_text       // текст письма
   );
```

Параметры

subject

[in] Заголовок письма.

some_text

[in] Тело письма.

Возвращаемое значение

true - если письмо поставлено в очередь на отсылку, иначе возвращает false.

Примечание

Отсылка может быть запрещена в настройках, также может быть не указан адрес электронной почты. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

При работе в тестере стратегий функция SendMail() не выполняется.

Пример:

```
//+------------------------------------------------------------------+
//|                                                     SendMail.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   SUBJECT   "Test SendMail"
#define   TEXT      "Text for SendMail() function"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- проверим разрешение на отправку электронной почты в терминале
   if(!TerminalInfoInteger(TERMINAL_EMAIL_ENABLED))
     {
      Print("Error. The client terminal does not have permission to send email message
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- отправляем письмо
   ResetLastError();
   if(!SendMail(SUBJECT, TEXT))
      Print("SendMail() failed. Error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

SendNotification

Посылает уведомление на мобильные терминалы, чьи MetaQuotes ID указаны в окне настроек на закладке "Уведомления".

```
bool  SendNotification(
   string  text          // текст сообщения
   );
```

Параметры

text

[in] Текст сообщения в уведомлении. Длина сообщения должна быть не более 255 символов.

Возвращаемое значение

true при успешной отправке уведомления из терминала, в случае неудачи возвращает false. При проверке после неудачной отправки уведомления GetLastError() может выдать одну из следующих ошибок:

- 4515 - ERR_NOTIFICATION_SEND_FAILED,
- 4516 - ERR_NOTIFICATION_WRONG_PARAMETER,
- 4517 - ERR_NOTIFICATION_WRONG_SETTINGS,
- 4518 - ERR_NOTIFICATION_TOO_FREQUENT.

Примечание

Для функции SendNotification() установлены жесткие ограничения по использованию: не более 2-х вызовов в секунду и не более 10 вызовов в минуту. Контроль за частотой использования осуществляется динамически, и функция может быть заблокирована при нарушении.

При работе в тестере стратегий функция SendNotification() не выполняется.

Пример:

```
//+------------------------------------------------------------------+
//|                                             SendNotification.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   MESSAGE   "Test Message"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- проверим разрешение на отправку уведомлений в терминале
   if(!TerminalInfoInteger(TERMINAL_NOTIFICATIONS_ENABLED))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("Error. The client terminal does not have permission to send notifications
      return;
     }
//--- отправляем уведомление
   ResetLastError();
   if(!SendNotification(MESSAGE))
      Print("SendNotification() failed. Error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

Глобальные переменные клиентского терминала

Группа функций, предназначенных для работы с глобальными переменными.

Не следует путать глобальные переменные клиентского терминала с переменными, объявленными на глобальном уровне mql5-программы.

Глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются. Обращением к глобальной переменной считается не только установка нового значения, но и чтение значения глобальной переменной.

Глобальные переменные клиентского терминала доступны одновременно из всех mql5-программ, запущенных на клиентском терминале.

Функция	Действие
GlobalVariableCheck	Проверяет существование глобальной переменной с указанным именем
GlobalVariableTime	Возвращает время последнего доступа к глобальной переменной
GlobalVariableDel	Удаляет глобальную переменную
GlobalVariableGet	Запрашивает значение глобальной переменной
GlobalVariableName	Возвращает имя глобальной переменной по порядковому номеру в списке глобальных переменных
GlobalVariableSet	Устанавливает новое значение глобальной переменной
GlobalVariablesFlush	Принудительно записывает содержимое всех глобальных переменных на диск
GlobalVariableTemp	Устанавливает новое значение глобальной переменной, которая существует только на время текущего сеанса работы терминала
GlobalVariableSetOnCondition	Устанавливает новое значение существующей глобальной переменной по условию
GlobalVariablesDeleteAll	Удаляет глобальные переменные с указанным префиксом в имени
GlobalVariablesTotal	Возвращает общее количество глобальных переменных

© 2000-2025, MetaQuotes Ltd.

GlobalVariableCheck

Проверяет существование глобальной переменной клиентского терминала.

```
bool  GlobalVariableCheck(
   string  name      // имя
   );
```

Параметры

name

[in] Имя глобальной переменной.

Возвращаемое значение

Возвращает значение true, если глобальная переменная существует, иначе возвращает false.

Примечание

Глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются.

Смотри также

GlobalVariableTime()

© 2000-2025, MetaQuotes Ltd.

GlobalVariableTime

Возвращает время последнего доступа к глобальной переменной.

```
datetime  GlobalVariableTime(
   string  name      // имя
   );
```

Параметры

name

[in] Имя глобальной переменной.

Возвращаемое значение

Возвращает время последнего доступа к указанной глобальной переменной. Обращение к переменной за значением, например, с помощью функций GlobalVariableGet() и GlobalVariableCheck(), также изменяет время последнего доступа. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются.

Смотри также

GlobalVariableCheck()

© 2000-2025, MetaQuotes Ltd.

GlobalVariableDel

Удаляет глобальную переменную клиентского терминала.

```
bool  GlobalVariableDel(
   string  name      // имя
   );
```

Параметры

name

[in] Имя глобальной переменной.

Возвращаемое значение

При успешном удалении функция возвращает true, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableGet

Возвращает значение существующей глобальной переменной клиентского терминала. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
double  GlobalVariableGet(
   string  name      // имя
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение глобальной переменной клиентского терминала помещается в приемную переменную, передаваемую по ссылке вторым параметром.

```
bool  GlobalVariableGet(
   string  name,              // имя
   double& double_var         // сюда примем значение глобальной переменной
   );
```

Параметры

name

[in] Имя глобальной переменной.

double_var

[out] Переменная типа double, принимающая значение, хранящееся в глобальной переменной клиентского терминала.

Возвращаемое значение

Значение существующей глобальной переменной или 0 в случае ошибки. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableName

Возвращает имя глобальной переменной по порядковому номеру.

```
string  GlobalVariableName(
   int  index      // номер в списке глобальных переменных
   );
```

Параметры

index

[in] Порядковый номер в списке глобальных переменных. Должен быть большим или равным 0 и меньшим, чем GlobalVariablesTotal() .

Возвращаемое значение

Имя глобальной переменной по порядковому номеру в списке глобальных переменных. Чтобы получить информацию об ошибке необходимо вызвать функцию функцию GetLastError().

Примечание

Глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableSet

Устанавливает новое значение глобальной переменной. Если переменная не существует, то система создает новую глобальную переменную.

```
datetime  GlobalVariableSet(
   string  name,       // имя
   double  value       // устанавливое значение
   );
```

Параметры

name

[in] Имя глобальной переменной.

value

[in] Новое числовое значение.

Возвращаемое значение

При успешном выполнении функция возвращает время последнего доступа, иначе 0. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Имя глобальной переменной не должно превышать 63 символа. Глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются.

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesFlush

Принудительная запись содержимого всех глобальных переменных на диск.

```
void  GlobalVariablesFlush();
```

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Терминал сам записывает все глобальные переменные при окончании работы, но при внезапном сбое работы компьютера данные могут потеряться. Данная функция позволяет самостоятельно управлять процессом сохранения глобальных переменных на случай нештатных ситуаций.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableTemp

Производит попытку создания временной глобальной переменной. Если переменная не существует, то система создает новую временную глобальную переменную.

```
bool  GlobalVariableTemp(
   string  name      // имя
   );
```

Параметры

name

[in] Имя временной глобальной переменной.

Возвращаемое значение

При успешном выполнении функция возвращает true, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Временные глобальные переменные существуют только во время работы клиентского терминала, после закрытия терминала они автоматически уничтожаются. При выполнении операции GlobalVariablesFlush() временные глобальные переменные на диск не записываются.

После создания временной глобальной переменной доступ к ней и ее модификация осуществляется точно так же, как и к обычной глобальной переменной клиентского терминала.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableSetOnCondition

Устанавливает новое значение существующей глобальной переменной, если текущее значение переменной равно значению третьего параметра check_value. Если переменной не существует, функция сгенерирует ошибку ERR_GLOBALVARIABLE_NOT_FOUND (4501) и вернет false.

```
bool  GlobalVariableSetOnCondition(
   string  name,             // имя
   double  value,            // значение при выполнении условия
   double  check_value       // проверяемое условие
   );
```

Параметры

name

[in] Имя глобальной переменной.

value

[in] Новое значение.

check_value

[in] Значение для проверки текущего значения глобальной переменной.

Возвращаемое значение

При успешном выполнении функция возвращает true, иначе false. Для получения информации об ошибке необходимо вызвать функцию GetLastError(). Если текущее значение глобальной переменной отличается от check_value, функция вернет false.

Примечание

Функция обеспечивает атомарный доступ к глобальной переменной, поэтому она может быть использована для организации мьютекса при взаимодействии нескольких одновременно работающих экспертов в пределах одного клиентского терминала.

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesDeleteAll

Удаляет глобальные переменные клиентского терминала.

```
int  GlobalVariablesDeleteAll(
   string     prefix_name=NULL,     // все глобальные переменные, чьи имена начинаются
   datetime   limit_data=0          // все глобальные переменные, которые изменялись р
   );
```

Параметры

prefix_name=NULL

[in] Префикс имени удаляемых глобальных переменных. Если указан префикс NULL либо пустая строка, то под критерий удаления соответствуют все глобальные переменные, соответствующие критерию удаления по дате

limit_data=0

[in] Дата для отбора глобальных переменных по времени последней модификации. Удаляются глобальные переменные, которые изменялись ранее указанной даты. Если параметр равен нулю, то удаляются все глобальные переменные, соответствующие первому критерию (по префиксу).

Возвращаемое значение

Количество удаленных переменных.

Примечание

Если оба параметра равны нулю (prefix_name=NULL и limit_data=0), то удаляются все глобальные переменные терминала. Если указаны оба параметра, то удаляются глобальные переменные, соответствующие одновременно каждому из указанных параметров.

Глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются.

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesTotal

Возвращает общее количество глобальных переменных клиентского терминала.

```
int  GlobalVariablesTotal();
```

Возвращаемое значение

Количество глобальных переменных.

Примечание

Глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются. Обращением к глобальной переменной считается не только установка нового значения, но и чтение значения глобальной переменной.

© 2000-2025, MetaQuotes Ltd.

Файловые операции

Группа функций для работы с файлами.

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы".

Существует два каталога (с подкаталогами), в которых могут располагаться рабочие файлы:

- папка_данных_терминала\MQL5\FILES\ (выберите для просмотра в терминале пункт меню "Файл"-"Открыть каталог данных");
- общая папка всех установленных на компьютере терминалов - обычно расположена в каталоге C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common\Files.

Программным путем можно получить наименования этих каталогов с помощью функции TerminalInfoString(), используя перечисление ENUM_TERMINAL_INFO_STRING:

```
//--- Папка, в которой хранятся данные терминала
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- Общая папка всех клиентских терминалов
   string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

Работа с файлами из других каталогов пресекается.

Если файл открывается для записи с помощью функции FileOpen(), то все указанные в пути подпапки будут созданы в случае их отсутствия.

Файловые функции позволяют работать с так называемыми "именованными каналами". Для этого достаточно вызвать функцию FileOpen() с соответствующими параметрами.

Функция	Действие
FileSelectDialog	Создает диалог открытия/создания файла или папки
FileFindFirst	Начинает перебор файлов в соответствующей директории в соответствии с указанным фильтром
FileFindNext	Продолжает поиск, начатый функцией FileFindFirst()
FileFindClose	Закрывает хэндл поиска
FileOpen	Открывает файл с указанным именем и указанными флагам
FileDelete	Удаляет указанный файл
FileFlush	Сброс на диск всех данных, оставшихся в файловом буфере ввода-вывода
FileGetInteger	Получает целочисленное свойство файла
FileIsEnding	Определяет конец файла в процессе чтения
FileIsLineEnding	Определяет конец строки в текстовом файле в процессе чтения
FileClose	Закрывает ранее открытый файл

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
FileIsExist	Проверяет существование файла
FileCopy	Копирует исходный файл из локальной или общей папки в другой файл
FileMove	Перемещает или переименовывает файл
FileReadArray	Читает массивы любых типов, кроме строковых (может быть массив структур, не содержащих строки и динамические массивы), из бинарного файла с текущего положения файлового указателя
FileReadBool	Читает из файла типа CSV строку от текущего положения до разделителя (либо до конца текстовой строки) и преобразует прочитанную строку в значение типа bool
FileReadDatetime	Читает из файла типа CSV строку одного из форматов: "YYYY.MM.DD HH:MM:SS", "YYYY.MM.DD" или "HH:MM:SS" - и преобразует ее в значение типа datetime
FileReadDouble	Читает число двойной точности с плавающей точкой (double) из бинарного файла с текущего положения файлового указателя
FileReadFloat	Читает из текущего положения файлового указателя значение типа float
FileReadInteger	Читает из бинарного файла значение типа int, short или char в зависимости от указанной длины в байтах
FileReadLong	Читает из текущего положения файлового указателя значение типа long
FileReadNumber	Читает из файла типа CSV строку от текущего положения до разделителя (либо до конца текстовой строки) и преобразует прочитанную строку в значение типа double
FileReadString	Читает из файла строку с текущего положения файлового указателя
FileReadStruct	Считывает из бинарного файла содержимое в структуру, переданную в качестве параметра
FileSeek	Перемещает положение файлового указателя на указанное количество байт относительно указанного положения
FileSize	Возвращает размер соответствующего открытого файла
FileTell	Возвращает текущее положение файлового указателя соответствующего открытого файла
FileWrite	Записывает данные в файл типа CSV или TXT
FileWriteArray	Записывает в файл типа BIN массивы любых типов, кроме строковых

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
FileWriteDouble	Записывает в бинарный файл значение параметра типа double с текущего положения файлового указателя
FileWriteFloat	Записывает в бинарный файл значение параметра типа float с текущего положения файлового указателя
FileWriteInteger	Записывает в бинарный файл значение параметра типа int с текущего положения файлового указателя
FileWriteLong	Записывает в бинарный файл значение параметра типа long с текущего положения файлового указателя
FileWriteString	Записывает в файл типа BIN или TXT значение параметра типа string с текущего положения файлового указателя
FileWriteStruct	Записывает в бинарный файл содержимое структуры, переданной в качестве параметра, с текущего положения файлового указателя
FileLoad	Считывает всё содержимое указанного бинарного файла в переданный массив числовых типов или простых структур
FileSave	Записывает в бинарный файл все элементы массива, переданного в качестве параметра
FolderCreate	Создает директорию в папке Files (в зависимости от значения common_flag)
FolderDelete	Удаляет указанную директорию. Если папка не пуста, то она не может быть удалена
FolderClean	Удаляет все файлы в указанной папке

© 2000-2025, MetaQuotes Ltd.

FileSelectDialog

Создает диалог открытия/создания файла или папки.

```
int  FileSelectDialog(
   string   caption,              // заголовок окна
   string   initial_dir,          // начальная папка
   string   filter,               // фильтр расширений
   uint     flags,                // комбинация флагов
   string&  filenames[],          // массив с именами файлов
   string   default_filename      // имя файла по умолчанию
   );
```

Параметры

caption

[in] Заголок диалогового окна.

initial_dir

[in] Имя начальной папки относительно папки MQL5\Files, содержимое которой будет показано в диалоговом окне. Если значение равно NULL, то в диалоге будет показана папка MQL5\Files.

filter

[in] Фильтр расширений файлов, которые будут показаны в диалоговом окне для выбора. Файлы других форматов будут скрыты.

flags

[in] Комбинация флагов, определяющая режим диалогового окна. Флаги определены следующим образом:
FSD_WRITE_FILE - диалог открытия файла;
FSD_SELECT_FOLDER - разрешает выбирать только папки;
FSD_ALLOW_MULTISELECT - разрешает выбирать несколько файлов;
FSD_FILE_MUST_EXIST - выбранные файлы должны существовать;
FSD_COMMON_FOLDER - файл расположен в общей папке всех клиентских терминалов \Terminal\Common\Files.

filenames[]

[out] Массив строк, в который будут помещены имена выбранных файлов/папок.

default_filename

[in] Имя файла/папки по умолчанию. Если задано, то это имя автоматически подставляется в диалог открытия и возвращается в массиве *filenames[]* при тестировании.

Возвращаемое значение

В случае успешного завершения функция вернет количество выбранных файлов, имена которых можно получить в *filenames[]*. Если пользователь не выбрал файл и закрыл диалог, то функция вернёт 0. В случае неудачного выполнения вернётся значение меньше 0, код ошибки можно получить с помощью GetLastError().

© 2000-2025, MetaQuotes Ltd.

Примечание

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы", а именно за пределами папки MQL5\Files.

Имя начальной папки *initial_dir* ищется в папке клиентского терминала в подпапке MQL5\Files (или каталог_агента_тестирования\MQL5\Files в случае тестирования). Если среди флагов указан FSD_COMMON_FOLDER, то начальная папка ищется в общей папке всех клиентских терминалов \Terminal\Common\Files.

Параметр *filter* указывает допустимые файлы и должен быть задан в формате "<описание 1>|<расширение 1>|<описание 2>|<расширение 2>...". Например, "Text files (*.txt)|*.txt|All files (*.*)|*.*", при этом первое расширение "Text files (*.txt)|*.txt" будет выбрано как тип файла по умолчанию.
Если *filter*=NULL, то маской выбора файлов в диалоговом окне будет "All Files (*.*)|*.*|"

Если задан параметр *default_filename*, то во время невизуального тестирования вызов FileSelectDialog() вернёт 1, а само значение *default_filename* будет скопировано в массив *filenames[]*.

Функция запрещена в пользовательских индикаторах, так как вызов FileSelectDialog() приостанавливает работу потока исполнения на всё время ожидания ответа пользователя. А так как все индикаторы по каждому символу выполняются в едином потоке, то такая остановка сделает невозможной работу всех графиков на всех таймфреймах по данному символу.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- получим имена текстовых файлов для загрузки из общей папки клиентских терминалов
   string filenames[];
   if(FileSelectDialog("Выберите файлы для загрузки", NULL,
                       "Text files (*.txt)|*.txt|All files (*.*)|*.*",
                       FSD_ALLOW_MULTISELECT|FSD_COMMON_FOLDER, filenames, "data.txt")
     {
      //--- выведем имя каждого выбранного файла
      int total=ArraySize(filenames);
      for(int i=0; i<total; i++)
         Print(i, ": ", filenames[i]);
     }
   else
     {
     Print("Files not selected");
     }
//---
  }
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

FileOpen, FileIsExist, FileDelete, FileMove, FolderCreate, FolderDelete, FolderClean, Флаги открытия файлов

© 2000-2025, MetaQuotes Ltd.

FileFindFirst

Начинает перебор файлов и поддиректорий в соответствующей директории в соответствии с указанным фильтром.

```
long  FileFindFirst(
   const string   file_filter,          // строка - фильтр поиска
   string&        returned_filename,    // имя найденного файла или поддиректории
   int            common_flag=0         // определяет область поиска
   );
```

Параметры

file_filter

[in] Фильтр поиска. В фильтре может быть указана поддиректория (или последовательность вложенных поддиректорий) относительно директории \Files, в которой необходимо проводить перебор файлов.

returned_filename

[out] Возвращаемый параметр, куда в случае удачи помещается имя первого найденного файла или поддиректории. Возвращается только имя файла (включая расширение) без указания директорий и поддиректорий, независимо от того, указывались ли они в фильтре для поиска.

common_flag

[in] Флаг, определяющий местоположение файла. Если common_flag=FILE_COMMON, то файл находится в общей папке всех клиентских терминалов \Terminal\Common\Files. В противном случае файл находится в локальной папке.

Возвращаемое значение

Возвращает хэндл объекта поиска, который необходимо использовать для дальнейшего перебора файлов и поддиректорий функцией FileFindNext(), либо INVALID_HANDLE в случае, когда нет ни одного файла и поддиректории, соответствующего фильтру (в частном случае - директория пуста). После окончания поиска хэндл необходимо закрыть при помощи функции FileFindClose().

Примечание

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы".

Пример:

© 2000-2025, MetaQuotes Ltd.

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- фильтр
input string InpFilter="Dir1\\*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   string int_dir="";
   int    i=1,pos=0,last_pos=-1;
//--- ищем последний бэк-слеш
   while(!IsStopped())
     {
      pos=StringFind(InpFilter,"\\",pos+1);
      if(pos>=0)
         last_pos=pos;
      else
         break;
     }
//--- в фильтре присутствует имя папки
   if(last_pos>=0)
      int_dir=StringSubstr(InpFilter,0,last_pos+1);
//--- получение хэндла поиска в корне локальной папки
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- проверим, успешно ли отработала функция FileFindFirst()
   if(search_handle!=INVALID_HANDLE)
     {
      //--- в цикле проверим являются ли переданные строки именами файлов или директор
      do
        {
         ResetLastError();
         //--- если это файл, то функция вернет true, а если директория, то функция ге
         FileIsExist(int_dir+file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- закрываем хэндл поиска
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Смотри также

FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

FileFindNext

Продолжает поиск, начатый функцией FileFindFirst().

```
bool  FileFindNext(
   long      search_handle,          // handle поиска
   string&   returned_filename       // имя найденного файла или поддиректории
   );
```

Параметры

search_handle

[in] Хэндл поиска, полученный функцией FileFindFirst().

returned_filename

[out] Имя следующего найденного файла или поддиректории. Возвращается только имя файла (включая расширение) без указания директорий и поддиректорий, независимо от того, указывались ли они в фильтре для поиска.

Возвращаемое значение

В случае удачи возвращает true, иначе false.

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- фильтр
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- получение хэндла поиска в корне локальной папки
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- проверим, успешно ли отработала функция FileFindFirst()
   if(search_handle!=INVALID_HANDLE)
     {
      //--- в цикле проверим являются ли переданные строки именами файлов или директор
      do
        {
         ResetLastError();
         //--- если это файл, то функция вернет true, а если директория, то функция ге
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- закрываем хэндл поиска
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Смотри также

FileFindFirst, FileFindClose

© 2000-2025, MetaQuotes Ltd.

FileFindClose

Закрывает хэндл поиска.

```
void  FileFindClose(
   long  search_handle      // handle поиска
   );
```

Параметры

search_handle

[in] Хэндл поиска, полученный функцией FileFindFirst().

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Функцию необходимо вызывать, чтобы освобождать системные ресурсы.

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- фильтр
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- получение хэндла поиска в корне локальной папки
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- проверим, успешно ли отработала функция FileFindFirst()
   if(search_handle!=INVALID_HANDLE)
     {
      //--- в цикле проверим являются ли переданные строки именами файлов или директор
      do
        {
         ResetLastError();
         //--- если это файл, то функция вернет true, а если директория, то функция ге
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==5018 ? "Directory" : "File"
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- закрываем хэндл поиска
      FileFindClose(search_handle);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      Print("Files not found!");
  }
```

Смотри также

FileFindFirst, FileFindNext

© 2000-2025, MetaQuotes Ltd.

FileIsExist

Проверяет существование файла.

```
bool  FileIsExist(
   const string  file_name,        // имя файла
   int           common_flag=0     // зона поиска
   );
```

Параметры

file_name

[in] Имя проверяемого файла.

common_flag=0

[in] Флаг, определяющий местоположение файла. Если common_flag=FILE_COMMON, то файл находится в общей папке всех клиентских терминалов \Terminal\Common\Files. В противном случае файл находится в локальной папке.

Возвращаемое значение

Возвращает true, если указанный файл существует.

Примечание

Проверяемый файл может оказаться поддиректорией. В этом случае функция FileIsExist() возвратит false, а в переменную _LastError будет записана ошибка 5018 - "Это не файл, а директория" (смотрите пример для функции FileFindFirst).

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы".

Если common_flag=FILE_COMMON, то функция ищет указанный файл в общей папке всех клиентских терминалов \Terminal\Common\Files, в противном случае функция ищет файл в локальной папке (MQL5\Files или MQL5\Tester\Files в случае тестирования).

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- дата для старых файлов
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;       // переменная для хранения имен файлов
   string   filter="*.txt"; // фильтр для поиска файлов
   datetime create_date;     // дата создания файла
   string   files[];         // список имен файлов
   int      def_size=25;     // размер массива по умолчанию
```

© 2000-2025, MetaQuotes Ltd.

```
   int      size=0;         // количество файлов
//--- выдели память для массива
   ArrayResize(files,def_size);
//--- получение хэндла поиска в корне локальной папки
   long search_handle=FileFindFirst(filter,file_name);
//--- проверим, успешно ли отработала функция FileFindFirst()
   if(search_handle!=INVALID_HANDLE)
     {
      //--- в цикле перебираем файлы
      do
        {
         files[size]=file_name;
         //--- увеличим размер массива
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- сбрасываем значение ошибки
         ResetLastError();
         //--- получим дату создания файла
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- проверим, старый ли файл
         if(create_date<InpFilesDate)
           {
            PrintFormat("Файл %s удален!",file_name);
            //--- удаляем старый файл
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- закрываем хэндл поиска
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- проверим какие из файлов остались
   PrintFormat("Результаты:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("Файл %s существует!",files[i]);
      else
         PrintFormat("Файл %s удален!",files[i]);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

Смотри также

FileFindFirst

© 2000-2025, MetaQuotes Ltd.

FileOpen

Функция открывает файл с указанным именем и указанными флагами.

```
int  FileOpen(
   string  file_name,           // имя файла
   int     open_flags,          // комбинация флагов
   short   delimiter='\t',      // разделитель
   uint    codepage=CP_ACP      // кодовая страница
   );
```

Параметры

file_name

[in] Имя открываемого файла, может содержать подпапки. Если файл открывается для записи, то указанные подпапки будут созданы в случае их отсутствия.

open_flags

[in] комбинация флагов, определяющая режим работы с файлом. Флаги определены следующим образом:
FILE_READ файл открывается для чтения
FILE_WRITE файл открывается для записи
FILE_BIN двоичный режим чтения-записи (без преобразования из строки и в строку)
FILE_CSV файл типа csv (все записанные элементы преобразуются к строкам соответствующего типа, unicode или ansi, и разделяются разделителем)
FILE_TXT простой текстовый файл (тот же csv, однако разделитель не принимается во внимание)
FILE_ANSI строки типа ANSI (однобайтовые символы)
FILE_UNICODE строки типа UNICODE (двухбайтовые символы)
FILE_SHARE_READ совместный доступ по чтению со стороны нескольких программ
FILE_SHARE_WRITE совместный доступ по записи со стороны нескольких программ
FILE_COMMON расположение файла в общей папке всех клиентских терминалов \Terminal\Common\Files.

delimiter='\t'

[in] значение, используемое в качестве разделителя в txt или csv-файле. Если для csv-файла разделитель не указан, то по умолчанию используется символ табуляции. Если для txt-файла разделитель не указан, то никакой разделитель не используется. Если в качестве разделителя явно задано значение 0, то никакой разделитель не используется.

codepage=CP_ACP

[in] Значение кодовой страницы. Для наиболее употребимых кодовых страниц предусмотрены соответствующие константы.

Возвращаемое значение

В случае успешного открытия функция возвращает хэндл файла, который затем используется для доступа к данным файла. В случае неудачи возвращает INVALID_HANDLE.

Примечание

© 2000-2025, MetaQuotes Ltd.

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы".

Если файл требуется прочитать в определенной кодировке (указан параметр codepage со значением кодовой страницы), то необходимо обязательно выставить флаг FILE_ANSI. Без указания флага FILE_ANSI чтение текстового файла будет происходить в Юникоде без какого-либо преобразования.

Файл открывается в папке клиентского терминала в подпапке MQL5\Files (или каталог_агента_тестирования\MQL5\Files в случае тестирования). Если среди флагов указан FILE_COMMON, то файл открывается в общей папке всех клиентских терминалов \Terminal\Common\Files.

Можно открывать "именованные каналы" по следующим правилам:

- Имя канала - строка, которая должна иметь вид: "\\servername\pipe\pipename", где servername - имя сервера в сети, а pipename - имя канала. Если каналы используются на одном и том же компьютере, имя сервера может быть опущено, но вместо него нужно поставить точку: "\\.\pipe\pipename". Клиент, который пытается соединиться с каналом, должен знать его имя.
- Необходимо вызывать FileFlush() и FileSeek() на начало файла между последовательными операциями чтения из канала и записи в канал.

В приведенных строках используется специальный символ обратная косая черта '\', поэтому при написании имени в MQL5 программе '\' необходимо удваивать, то есть вышеприведенный пример написать в коде как "\\\\servername\\pipe\\pipename".

Более подробно о работе с именованными каналами можно прочитать в статье "Связь с MetaTrader 5 через именованные каналы без применения DLL"

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- неправильный способ открытия файла
   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("Неудачная попытка открыть файл по абсолютному пути");
      Print("Код ошибки ",GetLastError());
     }
//--- правильный способ работы в "файловой песочнице"
   ResetLastError();
   filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
   if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
```

© 2000-2025, MetaQuotes Ltd.

```
      FileClose(filehandle);
      Print("FileOpen OK");
     }
   else Print("Операция FileOpen неудачна, ошибка ",GetLastError());
//--- еще один пример с созданием вложенной директории в 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("Файл должен быть создан в папке "+terminal_data_path+"\\"+subfolder);
     }
   else Print("Операция FileOpen неудачна, ошибка ",GetLastError());
  }
```

Смотри также

Использование кодовой страницы, FileFindFirst, FolderCreate, Флаги открытия файлов

© 2000-2025, MetaQuotes Ltd.

FileClose

Закрытие файла, ранее открытого функцией FileOpen().

```
void  FileClose(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Нет возвращаемого значения.

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string InpFileName="file.txt";     // имя файла
input string InpDirectoryName="Data";    // имя директории
input int    InpEncodingType=FILE_ANSI; // ANSI=32 или UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- распечатаем путь к папке в которой будем работать
   PrintFormat("Работаем в папке %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
//--- сбросим значение ошибки
   ResetLastError();
//--- откроем файл для чтения (если файл не существует, то произойдет ошибка)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //--- распечатаем содержимое файла
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- закрываем файл
      FileClose(file_handle);
     }
   else
      PrintFormat("Ошибка, код = %d",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

FileCopy

Копирует исходный файл из локальной или общей папки в другой файл.

```
bool  FileCopy(
   const string  src_file_name,      // имя файла источника
   int           common_flag,        // расположение файла
   const string  dst_file_name,      // имя файла назначения
   int           mode_flags          // способ доступа и/или расположение файла
   );
```

Параметры

src_file_name

[in] Имя файла для копирования.

common_flag

[in] Флаг, определяющий местоположение файла. Если common_flag=FILE_COMMON, то файл находится в общей папке всех клиентских терминалов \Terminal\Common\Files. В противном случае файл находится в локальной папке (например, *common_flag=0*).

dst_file_name

[in] Имя результатного файла.

mode_flags

[in] Флаги доступа. Параметр может содержать только 2 флага: FILE_REWRITE и/или FILE_COMMON - остальные флаги игнорируются. Если файл уже существует и при этом не был указан флаг FILE_REWRITE, то файл не будет переписан, и функция возвратит false.

Возвращаемое значение

В случае неудачи функция возвращает false.

Примечание

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы".

Если новый файл существовал, то копирование будет производиться в зависимости от наличия флага FILE_REWRITE в значении параметра mode_flags.

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string InpSrc="source.txt";       // источник
input string InpDst="destination.txt";  // копия
input int    InpEncodingType=FILE_ANSI; // ANSI=32 или UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- отобразим содержимое источника (он должен существовать)
   if(!FileDisplay(InpSrc))
      return;
//--- проверим, существует ли уже файл копии (не обязан быть создан)
   if(!FileDisplay(InpDst))
     {
      //--- файл копии не существует, копирование без флага FILE_REWRITE (правильное к
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
   else
     {
      //--- файл копии уже существует, попробуем скопировать без флага FILE_REWRITE (к
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
      //--- содержимое файла InpDst останется прежним
      FileDisplay(InpDst);
      //--- скопируем еще раз с флагом FILE_REWRITE (правильное копирование при сущест
      if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
//--- получили копию файла InpSrc
   FileDisplay(InpDst);
  }
//+------------------------------------------------------------------+
//| Чтение содержимого файла                                         |
//+------------------------------------------------------------------+
bool FileDisplay(const string file_name)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- откроем файл
   int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- отобразим в цикле содержимое файла
      Print("+----------------------+");
      PrintFormat("File name = %s",file_name);
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      Print("+----------------------+");
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- закроем файл
      FileClose(file_handle);
      return(true);
     }
//--- не удалось открыть файл
   PrintFormat("%s is not opened, error = %d",file_name,GetLastError());
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileDelete

Удаляет указанный файл в локальной папке клиентского терминала.

```
bool  FileDelete(
   const string  file_name,      // имя удаляемого файла
   int           common_flag=0   // местоположение удаляемого файла
   );
```

Параметры

file_name

[in] Имя файла.

common_flag=0

[in] Флаг, определяющий местоположение файла. Если common_flag=FILE_COMMON, то файл находится в общей папке всех клиентских терминалов \Terminal\Common\Files. В противном случае файл находится в локальной папке.

Возвращаемое значение

В случае неудачи функция возвращает false.

Примечание

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы".

Удаляет указанный файл в локальной папке клиентского терминала (MQL5\Files или MQL5\Tester\Files в случае тестирования). Если же указан common_flag=FILE_COMMON, то функция удаляет файл из общей папки всех клиентских терминалов.

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- дата для старых файлов
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // переменная для хранения имен файлов
   string   filter="*.txt"; // фильтр для поиска файлов
   datetime create_date;    // дата создания файла
   string   files[];        // список имен файлов
   int      def_size=25;    // размер массива по умолчанию
   int      size=0;         // количество файлов
//--- выдели память для массива
   ArrayResize(files,def_size);
//--- получение хэндла поиска в корне локальной папки
```

© 2000-2025, MetaQuotes Ltd.

```
   long search_handle=FileFindFirst(filter,file_name);
//--- проверим, успешно ли отработала функция FileFindFirst()
   if(search_handle!=INVALID_HANDLE)
     {
      //--- в цикле перебираем файлы
      do
        {
         files[size]=file_name;
         //--- увеличим размер массива
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- сбрасываем значение ошибки
         ResetLastError();
         //--- получим дату создания файла
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- проверим, старый ли файл
         if(create_date<InpFilesDate)
           {
            PrintFormat("Файл %s удален!",file_name);
            //--- удаляем старый файл
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- закрываем хэндл поиска
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- проверим какие из файлов остались
   PrintFormat("Результаты:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("Файл %s существует!",files[i]);
      else
         PrintFormat("Файл %s удален!",files[i]);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

FileMove

Перемещает файл из локальной или общей папки в другую папку.

```
bool  FileMove(
   const string  src_file_name,     // имя файла для операции перемещения
   int           common_flag,       // расположение файла
   const string  dst_file_name,     // имя файла назначения
   int           mode_flags         // способ доступа и/или расположение файла
   );
```

Параметры

src_file_name

[in] Имя файла для перемещения/переименования.

common_flag

[in] Флаг, определяющий местоположение файла. Если common_flag=FILE_COMMON, то файл находится в общей папке всех клиентских терминалов \Terminal\Common\Files. В противном случае файл находится в локальной папке.(*common_flag=0*).

dst_file_name

[in] Имя результатного файла.

mode_flags

[in] Флаги доступа. Параметр может содержать только 2 флага: FILE_REWRITE и/или FILE_COMMON - остальные флаги игнорируются. Если файл уже существует и при этом не был указан флаг FILE_REWRITE, то файл не будет переписан, и функция возвратит false.

Возвращаемое значение

В случае неудачи функция возвращает false.

Примечание

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы".

Если новый файл существовал, то копирование будет производиться в зависимости от наличия флага FILE_REWRITE в значении параметра mode_flags.

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   string local=TerminalInfoString(TERMINAL_DATA_PATH);
   string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- получим пути к файлам
   string src_path;
   string dst_path;
   StringConcatenate(src_path,InpSrcDirectory,"//",InpSrcName);
   StringConcatenate(dst_path,InpDstDirectory,"//",InpDstName);
//--- проверим, существуют ли файл источника (если нет - выход)
   if(FileIsExist(src_path))
      PrintFormat("%s file exists in the %s\\Files\\%s folder",InpSrcName,local,InpSrc
   else
     {
      PrintFormat("Error, %s source file not found",InpSrcName);
      return;
     }
//--- проверим, существует ли уже файл результата
   if(FileIsExist(dst_path,FILE_COMMON))
     {
      PrintFormat("%s file exists in the %s\\Files\\%s folder",InpDstName,common,InpDs
      //--- файл существует, перемещение нужно проводить с флагом FILE_REWRITE
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
   else
     {
      PrintFormat("%s file does not exist in the %s\\Files\\%s folder",InpDstName,comm
      //--- файл не существует, перемещение нужно проводить без флага FILE_REWRITE
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
//--- теперь файл перемещен, проверим это
   if(FileIsExist(dst_path,FILE_COMMON) && !FileIsExist(src_path,0))
      Print("Success!");
   else
      Print("Error!");
  }
```

Смотри также

FileIsExist

© 2000-2025, MetaQuotes Ltd.

FileFlush

Сброс на диск всех данных, оставшихся в файловом буфере ввода-вывода.

```
void  FileFlush(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Нет возвращаемого значения.

Примечание

При выполнении операции записи в файл физически данные могут оказаться в нем только через некоторое время. Для того чтобы данные сразу же сохранились в файле, нужно использовать функцию FileFlush(). Если не использовать функцию, то часть данных, еще не попавших на диск, принудительно записывается туда только при закрытии файла функцией FileClose().

Функцию необходимо использовать тогда, когда записываемые данные представляют собой определенную ценность. При этом стоит учитывать, что частый вызов функции может сказаться на скорости работы программы.

Функцию FileFlush() необходимо вызывать между операциями чтения из файла и записи в файл.

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- имя файла для записи
input string InpFileName="example.csv"; // имя файла
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- откроем файл
   int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- запишем данные в файл
      for(int i=0;i<1000;i++)
        {
         //--- вызовем функцию записи
         FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble(Symbol(),SYMBOL_BID),Sy
         //--- сбрасываем данные на диск на каждой 128 итерации
```

© 2000-2025, MetaQuotes Ltd.

```
         if((i & 127)==127)
           {
            //--- теперь данные будут находиться в файле, и при критической ошибке тер
            FileFlush(file_handle);
            PrintFormat("i = %d, OK",i);
           }
         //--- задержка в 0.01 секунды
         Sleep(10);
        }
      //--- закрываем файл
      FileClose(file_handle);
     }
   else
      PrintFormat("Ошибка, код = %d",GetLastError());
  }
```

Смотри также

FileClose

© 2000-2025, MetaQuotes Ltd.

FileGetInteger

Получает целочисленное свойство файла. Существует 2 варианта функции.

1. Получение свойств по хэндлу файла.

```
long  FileGetInteger(
   int                          file_handle,    // хэндл файла
   ENUM_FILE_PROPERTY_INTEGER   property_id     // идентификатор свойства
   );
```

2. Получение свойств по имени файла.

```
long  FileGetInteger(
   const string                 file_name,             // имя файла
   ENUM_FILE_PROPERTY_INTEGER   property_id,           // идентификатор свойства
   bool                         common_folder=false    // файл просматривается в локальн
   );                                                  // или в общей папке всех термина
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

file_name

[in] Имя файла.

property_id

[in] Идентификатор свойства файла. Значение может быть одним из значений перечисления ENUM_FILE_PROPERTY_INTEGER. Если используется второй вариант функции, то можно получать значения только следующих свойств: FILE_EXISTS, FILE_CREATE_DATE, FILE_MODIFY_DATE, FILE_ACCESS_DATE и FILE_SIZE.

common_folder=false

[in] Указывает на местоположение файла. Если параметр равен false, то просматривается каталог данных терминала, в противном случае предполагается, что файл находится в общей папке всех клиентских терминалов \Terminal\Common\Files (FILE_COMMON).

Возвращаемое значение

Значение свойства. В случае ошибки функция возвращает -1, для получения кода ошибки необходимо вызвать функцию GetLastError().

Если при получении свойств по имени будет указан каталог, то функция в любом случае выставит ошибку 5018 (ERR_MQL_FILE_IS_DIRECTORY), при этом возвращаемое значение будет корректным.

Примечание

Функция всегда изменяет код ошибки. При успешном завершении код ошибки сбрасывается в ноль.

Пример:

© 2000-2025, MetaQuotes Ltd.

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string InpFileName="data.csv";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string path=InpDirectoryName+"//"+InpFileName;
   long   l=0;
//--- откроем файл
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_CSV);
   if(handle!=INVALID_HANDLE)
     {
      //--- распечатаем всю информацию о файле
      Print(InpFileName," file info:");
      FileInfo(handle,FILE_EXISTS,l,"bool");
      FileInfo(handle,FILE_CREATE_DATE,l,"date");
      FileInfo(handle,FILE_MODIFY_DATE,l,"date");
      FileInfo(handle,FILE_ACCESS_DATE,l,"date");
      FileInfo(handle,FILE_SIZE,l,"other");
      FileInfo(handle,FILE_POSITION,l,"other");
      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");
      //--- закроем файл
      FileClose(handle);
     }
   else
      PrintFormat("%s file is not opened, ErrorCode = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------+
//| Отображение значения свойства файла                              |
//+------------------------------------------------------------------+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
              long l,const string type)
  {
//--- получим значение свойства
   ResetLastError();
   if((l=FileGetInteger(handle,id))!=-1)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- значение получено, отобразим его в правильном формате
      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());
  }
```

Смотри также

Файловые операции, Свойства файлов

© 2000-2025, MetaQuotes Ltd.

FileIsEnding

Определяет конец файла в процессе чтения.

```
bool  FileIsEnding(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Функция возвращает true в случае, если в процессе чтения или перемещения файлового указателя достигнут конец файла.

Примечание

Для определения конца файла, функция пытается провести чтение следующей строки из файла. Если ее не существует, то функция возвращает true, в противном случае false.

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string InpFileName="file.txt";     // имя файла
input string InpDirectoryName="Data";    // имя директории
input int    InpEncodingType=FILE_ANSI; // ANSI=32 или UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- распечатаем путь к папке в которой будем работать
   PrintFormat("Работаем в папке %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
//--- сбросим значение ошибки
   ResetLastError();
//--- откроем файл для чтения (если файл не существует, то произойдет ошибка)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //--- распечатаем содержимое файла
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- закрываем файл
      FileClose(file_handle);
     }
   else
      PrintFormat("Ошибка, код = %d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

FileIsLineEnding

Определяет конец строки в текстовом файле в процессе чтения.

```
bool  FileIsLineEnding(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Возвращает true в случае, если в процессе чтения txt или csv-файла достигнут конец строки (символы CR-LF).

Пример (используется файл, полученный в результате работы примера для функции FileWriteString)

```
//+------------------------------------------------------------------+
//|                                      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
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- параметры для чтения данных
input string InpFileName="RSI.csv";   // имя файла
input string InpDirectoryName="Data"; // имя директории
//--- индикаторные буферы
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- переменные перекупленности
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- переменные перепроданности
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- переменные размеров массивов по умолчанию
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- выделим память для массивов
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      double value;
      //--- читаем данные из файла
      while(!FileIsEnding(file_handle))
        {
         //--- читаем первое значение в строке
         value=FileReadNumber(file_handle);
         //--- читаем в разные массивы в зависимости от результата функции
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- закроем файл
      FileClose(file_handle);
      PrintFormat("Данные прочитаны, файл %s закрыт",InpFileName);
     }
   else
     {
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastErr
      return(INIT_FAILED);
     }
//--- привязка массивов
   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);
```

© 2000-2025, MetaQuotes Ltd.

```
//---- установка значений индикатора, которые не будут видимы на графике
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Чтение данных строки файла                                       |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- читаем пока не дойдем до конца строки или файла
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- сдвинем каретку, прочитав число
      if(flag)
         FileReadNumber(file_handle);
      //--- запоминаем текущую дату
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- при необходимости увеличим размер массива
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- проскочили первую итерацию
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- цикл для еще необработанных баров
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- по умолчанию 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- проверка, есть ли еще данные
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- если даты совпадают, то бар лежит в зоне перекупленности
            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 - красный цвет
               color_buff[i]=0;
               //--- увеличим счетчик
               ovb_ind=j+1;
               break;
              }
           }
      //--- проверка, есть ли еще данные
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- если даты совпадают, то бар лежит в зоне перепроданности
            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 - синий цвет
               color_buff[i]=1;
               //--- увеличим счетчик
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Обработчик события ChartEvent                                    |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//--- изменяем толщину индикатора в зависимости от масштаба
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Смотри также

FileWriteString

© 2000-2025, MetaQuotes Ltd.

FileReadArray

Читает массивы любых типов, кроме строковых (может быть массив структур, не содержащих строки и динамические массивы), из бинарного файла с текущего положения файлового указателя.

```
uint  FileReadArray(
   int     file_handle,               // handle файла
   void&   array[],                   // массив для записи
   int     start=0,                   // стартовая позиция для записи в массив
   int     count=WHOLE_ARRAY          // сколько читать
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

array[]

[out] Массив, куда данные будут загружены.

start=0

[in] Стартовая позиция для чтения из массива.

count=WHOLE_ARRAY

[in] Количество элементов для чтения.

Возвращаемое значение

Количество прочитанных элементов. По умолчанию, читает весь массив (count=WHOLE_ARRAY).

Примечание

Строковый массив может читаться только из файла типа TXT. При необходимости функция пытается увеличить размер массива.

Пример (используется файл, полученный в результате работы примера для функции FileWriteArray)

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Структура для хранения данных о ценах                            |
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // дата
   double            bid;  // цена бид
   double            ask;  // цена аск
  };
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- массив структуры
   prices arr[];
//--- путь к файлу
   string path=InpDirectoryName+"//"+InpFileName;
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- прочитаем все данные из файла в массив
      FileReadArray(file_handle,arr);
      //--- получим размер массива
      int size=ArraySize(arr);
      //--- распечатаем данные из массива
      for(int i=0;i<size;i++)
         Print("Date = ",arr[i].date," Bid = ",arr[i].bid," Ask = ",arr[i].ask);
      Print("Total data = ",size);
      //--- закрываем файл
      FileClose(file_handle);
     }
   else
      Print("File open failed, error ",GetLastError());
  }
```

Смотри также

Переменные, FileWriteArray

© 2000-2025, MetaQuotes Ltd.

FileReadBool

Читает из файла типа CSV строку от текущего положения до разделителя (либо до конца текстовой строки) и преобразует прочитанную строку в значение типа bool.

```
bool  FileReadBool(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Прочитанная строка может иметь значения "true", "false" или символьное представление целых чисел "0" или "1". Ненулевое значение преобразуется к логическому true. Функция возвращает полученное преобразованное значение.

Пример (используется файл, полученный в результате работы примера для функции FileWrite)

```
//+------------------------------------------------------------------+
//|                                              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
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- параметры для чтения данных
input string InpFileName="MACD.csv";  // имя файла
input string InpDirectoryName="Data"; // имя директории
//--- глобальные переменные
int      ind=0;       // индекс
double   upbuff[];    // индикаторный буфер стрелок вверх
```

© 2000-2025, MetaQuotes Ltd.

```
double   downbuff[];  // индикаторный буфер стрелок вниз
bool     sign_buff[]; // массив сигналов (true - покупка, false - продажа)
datetime time_buff[]; // массив времени наступления сигналов
int      size=0;      // размер массивов сигналов
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      //--- сначала прочитаем количество сигналов
      size=(int)FileReadNumber(file_handle);
      //--- выделим память для массивов
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- прочитаем данные из файла
      for(int i=0;i<size;i++)
        {
         //--- время сигнала
         time_buff[i]=FileReadDatetime(file_handle);
         //--- значение сигнала
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- закрываем файл
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
      return(INIT_FAILED);
     }
//--- привязка массивов
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- зададим код символа для отрисовки в PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- установка значений индикатора, которые не будут видимы на графике
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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(high,false);
   ArraySetAsSeries(low,false);
//--- цикл для еще необработанных баров
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- по умолчанию 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- проверка, есть ли еще данные
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- если даты совпадают, то используем значение из файла
            if(time[i]==time_buff[j])
              {
               //--- рисуем стрелку в зависимости от сигнала
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- увеличим счетчик
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Тип bool, FileWrite

© 2000-2025, MetaQuotes Ltd.

FileReadDatetime

Читает из файла типа CSV строку одного из форматов: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" или "HH:MI:SS" - и преобразует её в значение типа datetime.

```
datetime  FileReadDatetime(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Значение типа datetime.

Пример (используется файл, полученный в результате работы примера для функции FileWrite)

```
//+------------------------------------------------------------------+
//|                                        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
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- параметры для чтения данных
input string InpFileName="MACD.csv";  // имя файла
input string InpDirectoryName="Data"; // имя директории
//--- глобальные переменные
int      ind=0;        // индекс
double   upbuff[];     // индикаторный буфер стрелок вверх
double   downbuff[];   // индикаторный буфер стрелок вниз
bool     sign_buff[]; // массив сигналов (true - покупка, false - продажа)
```

© 2000-2025, MetaQuotes Ltd.

```
datetime time_buff[]; // массив времени наступления сигналов
int      size=0;      // размер массивов сигналов
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      //--- сначала прочитаем количество сигналов
      size=(int)FileReadNumber(file_handle);
      //--- выделим память для массивов
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- прочитаем данные из файла
      for(int i=0;i<size;i++)
        {
         //--- время сигнала
         time_buff[i]=FileReadDatetime(file_handle);
         //--- значение сигнала
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- закрываем файл
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
      return(INIT_FAILED);
     }
//--- привязка массивов
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- зададим код символа для отрисовки в PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- установка значений индикатора, которые не будут видимы на графике
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

© 2000-2025, 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(high,false);
   ArraySetAsSeries(low,false);
//--- цикл для еще необработанных баров
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- по умолчанию 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- проверка, есть ли еще данные
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- если даты совпадают, то используем значение из файла
            if(time[i]==time_buff[j])
              {
               //--- рисуем стрелку в зависимости от сигнала
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- увеличим счетчик
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Тип datetime, StringToTime, TimeToString, FileWrite

© 2000-2025, MetaQuotes Ltd.

FileReadDouble

Читает число двойной точности с плавающей точкой (double) из бинарного файла с текущего положения файлового указателя.

```
double  FileReadDouble(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Значение типа double.

Примечание

Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример (используется файл, полученный в результате работы примера для функции FileWriteDouble)

```
//+------------------------------------------------------------------+
//|                                          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
//---- plot Label1
#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
//--- параметры для чтения данных
input string InpFileName="MA.csv";     // имя файла
input string InpDirectoryName="Data"; // имя директории
//--- глобальные переменные
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- indicator buffer
```

© 2000-2025, MetaQuotes Ltd.

```
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- сначала прочитаем сколько всего в файле данных
      size=(int)FileReadDouble(file_handle);
      //--- выделим память для массивов
      ArrayResize(ma_buff,size);
      ArrayResize(time_buff,size);
      //--- прочитаем данные из файла
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadDouble(file_handle);
         ma_buff[i]=FileReadDouble(file_handle);
        }
      //--- закроем файл
      FileClose(file_handle);
      PrintFormat("Данные прочитаны, файл %s закрыт",InpFileName);
     }
   else
     {
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
      return(INIT_FAILED);
     }
//--- привязка массива к индикаторному буферу с индексом 0
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- установка значений индикатора, которые не будут видимы на графике
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- цикл для еще необработанных баров
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- по умолчанию 0
      buff[i]=0;
      //--- проверка, есть ли еще данные
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- если даты совпадают, то используем значение из файла
            if(time[i]==time_buff[j])
              {
               buff[i]=ma_buff[j];
               //--- увеличим счетчик
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Вещественные типы (double, float), StringToDouble, DoubleToString, FileWriteDouble

© 2000-2025, MetaQuotes Ltd.

FileReadFloat

Читает число одинарной точности с плавающей точкой (float) из текущей позиции бинарного файла.

```
float  FileReadFloat(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функциейFileOpen().

Возвращаемое значение

Значение типа float.

Примечание

Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример (используется файл, полученный в результате работы примера для функции FileWriteFloat)

```
//+------------------------------------------------------------------+
//|                                          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
//---- plot Label1
#property indicator_label1  "CloseLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- параметры для чтения данных
input string InpFileName="Close.bin"; // имя файла
input string InpDirectoryName="Data"; // имя директории
//--- глобальные переменные
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- indicator buffers
double   buff[];
```

© 2000-2025, MetaQuotes Ltd.

```
double   color_buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- выделим память для массивов
   ArrayResize(close_buff,def_size);
   ArrayResize(time_buff,def_size);
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- прочитаем данные из файла
      while(!FileIsEnding(file_handle))
        {
         //--- считываем значение времени и цены
         time_buff[size]=(datetime)FileReadDouble(file_handle);
         close_buff[size]=(double)FileReadFloat(file_handle);
         size++;
         //--- увеличим размеры массивов если они переполнены
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(close_buff,def_size);
            ArrayResize(time_buff,def_size);
           }
        }
      //--- закроем файл
      FileClose(file_handle);
      PrintFormat("Данные прочитаны, файл %s закрыт",InpFileName);
     }
   else
     {
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
      return(INIT_FAILED);
     }
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,buff,INDICATOR_DATA);
   SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- установка значений индикатора, которые не будут видимы на графике
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
//--- цикл для еще необработанных баров
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- по умолчанию 0
      buff[i]=0;
      color_buff[i]=0; // красный цвет по умолчанию
      //--- проверка, есть ли еще данные
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- если даты совпадают, то используем значение из файла
            if(time[i]==time_buff[j])
              {
               //--- получим цену
               buff[i]=close_buff[j];
               //--- если текущая цена больше предыдущий, то цвет синий
               if(buff[i-1]>buff[i])
                  color_buff[i]=1;
               //--- увеличим счетчик
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Вещественные типы (double, float), FileReadDouble, FileWriteFloat

© 2000-2025, MetaQuotes Ltd.

FileReadInteger

Читает из бинарного файла значение типа int, short или char в зависимости от указанной длины в байтах. Чтение производится с текущего положения файлового указателя.

```
int  FileReadInteger(
   int  file_handle,         // handle файла
   int  size=INT_VALUE       // размер целого типа в байтах
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

size=INT_VALUE

[in] Количество байт (до 4 включительно), которые нужно прочитать. Предусмотрены соответствующие константы: CHAR_VALUE=1, SHORT_VALUE=2 и INT_VALUE=4, таким образом функция может прочитать целое значение типа char, short или int.

Возвращаемое значение

Значение типа int. Результат этой функции необходимо явно приводить к целевому типу, то есть тому типу данных, который требуется прочитать. Так как возвращается значение типа int, то его можно спокойно преобразовать в любое целое значение. Файловый указатель перемещается на количество считываемых байт.

Примечание

При чтении менее 4-х байт, полученный результат всегда будет положительным. Если читается один или два байта, то знак числа можно точно определить путем явного приведения соответственно к типу char (1 байт) или типу short (2 байта). Получение знака для трехбайтового числа является нетривиальным, так как нет соответствующего базового типа.

Пример (используется файл, полученный в результате работы примера для функции FileWriteInteger)

```
//+------------------------------------------------------------------+
//|                                        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
//---- plot Label1
#property indicator_label1  "Trends"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- параметры для чтения данных
input string InpFileName="Trend.bin"; // имя файла
input string InpDirectoryName="Data"; // имя директории
//--- глобальные переменные
int      ind=0;
int      size=0;
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- выделим память для массива
   ArrayResize(time_buff,def_size);
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- вспомогательные переменные
      int    arr_size;
      uchar  arr[];
      //--- прочитаем данные из файла
      while(!FileIsEnding(file_handle))
        {
         //--- узнаем сколько символов использовано для записи времени
         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);
         //--- запомним значение времени
         time_buff[size]=StringToTime(CharArrayToString(arr));
         size++;
         //--- увеличим размеры массивов если они переполнены
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(time_buff,def_size);
           }
        }
      //--- закроем файл
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Данные прочитаны, файл %s закрыт",InpFileName);
     }
   else
     {
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
      return(INIT_FAILED);
     }
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- установка значений индикатора, которые не будут видимы на графике
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
//--- цикл для еще необработанных баров
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- по умолчанию 0
      buff[i]=0;
      //--- проверка, есть ли еще данные
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- если даты совпадают, то используем значение из файла
            if(time[i]==time_buff[j])
              {
               //--- получим цену
               buff[i]=close[i];
               //--- увеличим счетчик
               ind=j+1;
               break;
              }
```

© 2000-2025, MetaQuotes Ltd.

```
            }
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

IntegerToString, StringToInteger, Целые типы, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileReadLong

Читает целое число типа long (8 байт) из текущей позиции бинарного файла.

```
long  FileReadLong(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Значение типа long.

Пример (используется файл, полученный в результате работы примера для функции FileWriteLong)

```
//+------------------------------------------------------------------+
//|                                              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
//---- plot Label1
#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
//--- параметры для чтения данных
input string InpFileName="Volume.bin"; // имя файла
input string InpDirectoryName="Data";  // имя директории
//--- глобальные переменные
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для записи",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- сначала прочитаем сколько всего в файле данных
      size=(int)FileReadLong(file_handle);
      //--- выделим память для массивов
      ArrayResize(volume_buff,size);
      ArrayResize(time_buff,size);
      //--- прочитаем данные из файла
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadLong(file_handle);
         volume_buff[i]=FileReadLong(file_handle);
        }
      //--- закроем файл
      FileClose(file_handle);
      PrintFormat("Данные прочитаны, файл %s закрыт",InpFileName);
     }
   else
     {
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
      return(INIT_FAILED);
     }
//--- привязка массива к индикаторному буферу с индексом 0
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- установка значений индикатора, которые не будут видимы на графике
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ArraySetAsSeries(time,false);
//--- цикл для еще необработанных баров
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- по умолчанию 0
      buff[i]=0;
      //--- проверка, есть ли еще данные
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- если даты совпадают, то используем значение из файла
            if(time[i]==time_buff[j])
              {
               buff[i]=(double)volume_buff[j];
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Целые типы, FileReadInteger, FileWriteLong

© 2000-2025, MetaQuotes Ltd.

FileReadNumber

Читает из файла типа CSV строку от текущего положения до разделителя (либо до конца текстовой строки) и преобразует прочитанную строку в значение типа double.

```
double  FileReadNumber(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Значение типа double.

Пример (используется файл, полученный в результате работы примера для функции FileWriteString)

```
//+------------------------------------------------------------------+
//|                                          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
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- параметры для чтения данных
input string InpFileName="RSI.csv";   // имя файла
input string InpDirectoryName="Data"; // имя директории
//--- индикаторные буферы
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- переменные перекупленности
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- переменные перепроданности
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- переменные размеров массивов по умолчанию
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- выделим память для массивов
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      double value;
      //--- читаем данные из файла
      while(!FileIsEnding(file_handle))
        {
         //--- читаем первое значение в строке
         value=FileReadNumber(file_handle);
         //--- читаем в разные массивы в зависимости от результата функции
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- закроем файл
      FileClose(file_handle);
      PrintFormat("Данные прочитаны, файл %s закрыт",InpFileName);
     }
   else
     {
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastErr
      return(INIT_FAILED);
     }
//--- привязка массивов
   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);
```

© 2000-2025, MetaQuotes Ltd.

```
//---- установка значений индикатора, которые не будут видимы на графике
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Чтение данных строки файла                                       |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- читаем пока не дойдем до конца строки или файла
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- сдвинем каретку, прочитав число
      if(flag)
         FileReadNumber(file_handle);
      //--- запоминаем текущую дату
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- при необходимости увеличим размер массива
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- проскочили первую итерацию
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- цикл для еще необработанных баров
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- по умолчанию 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- проверка, есть ли еще данные
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- если даты совпадают, то бар лежит в зоне перекупленности
            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 - красный цвет
               color_buff[i]=0;
               //--- увеличим счетчик
               ovb_ind=j+1;
               break;
              }
           }
      //--- проверка, есть ли еще данные
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- если даты совпадают, то бар лежит в зоне перепроданности
            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 - синий цвет
               color_buff[i]=1;
               //--- увеличим счетчик
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Обработчик события ChartEvent                                    |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//--- изменяем толщину индикатора в зависимости от масштаба
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Смотри также

FileWriteString

© 2000-2025, MetaQuotes Ltd.

FileReadString

Читает из файла строку с текущего положения файлового указателя.

```
string  FileReadString(
   int  file_handle,     // handle файла
   int  length=-1        // длина строки
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

length=-1

[in] Количество символов для чтения.

Возвращаемое значение

Прочитанная строка(string).

Примечание

При чтении из bin-файла обязательно указывать длину читаемой строки. При чтении из txt-файла длину строки указывать не надо, будет прочитана строка от текущего положения до признака перевода строки "\r\n". При чтении из csv-файла длину строки также указывать не надо, будет прочитана строка от текущего положения до ближайшего разделителя либо до признака конца текстовой строки.

Если файл открыт с флагом FILE_ANSI, то прочитанная строка преобразовывается в Unicode.

Пример (используется файл, полученный в результате работы примера для функции FileWriteInteger)

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- параметры для чтения данных
input string InpFileName="Trend.bin"; // имя файла
input string InpDirectoryName="Data"; // имя директории
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- вспомогательные переменные
      int    str_size;
```

© 2000-2025, MetaQuotes Ltd.

```
      string str;
      //--- прочитаем данные из файла
      while(!FileIsEnding(file_handle))
        {
         //--- узнаем сколько символов использовано для записи времени
         str_size=FileReadInteger(file_handle,INT_VALUE);
         //--- прочитаем строку
         str=FileReadString(file_handle,str_size);
         //--- распечатаем строку
         PrintFormat(str);
        }
      //--- закроем файл
      FileClose(file_handle);
      PrintFormat("Данные прочитаны, файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
  }
```

Смотри также

Тип string, Преобразование данных, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileReadStruct

Считывает из бинарного файла содержимое в структуру, переданную в качестве параметра. Чтение производится с текущего положения файлового указателя.

```
uint  FileReadStruct(
   int           file_handle,        // handle файла
   const void&   struct_object,      // структура, куда происходит считывание
   int           size=-1             // размер структуры в байтах
   );
```

Параметры

file_handle

[in] Файловый описатель открытого бинарного файла.

struct_object

[out] Ссылка на объект указанной структуры. Структура не должна содержать строки, динамические массивы, виртуальные функции, а также указатели на объекты и функции.

size=-1

[in] Количество байт, которые нужно прочитать. Если размер не указан или указано большее количество байт, чем размер структуры, то используется точный размер указанной структуры.

Возвращаемое значение

В случае удачи функция возвращает количество прочитанных байт. Файловый указатель перемещается на это же количество байт.

Пример (используется файл, полученный в результате работы примера для функции FileWriteStruct)

```
//+------------------------------------------------------------------+
//|                                          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
//---- plot Label1
#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
//--- параметры для получения данных
input string  InpFileName="EURUSD.txt"; // имя файла
```

© 2000-2025, MetaQuotes Ltd.

```
input string  InpDirectoryName="Data";  // имя директории
//+------------------------------------------------------------------+
//| Структура для хранения данных свечи                              |
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // цена открытия
   double            close; // цена закрытия
   double            high;  // максимальная цена
   double            low;   // минимальная цена
   datetime          date;  // дата
  };
//--- индикаторные буферы
double       open_buff[];
double       close_buff[];
double       high_buff[];
double       low_buff[];
//--- глобальные переменные
candlesticks cand_buff[];
int          size=0;
int          ind=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int default_size=100;
   ArrayResize(cand_buff,default_size);
//--- откроем файл
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_P
      //--- прочитаем данные из файла
      while(!FileIsEnding(file_handle))
        {
         //--- запишем данные в массив
         FileReadStruct(file_handle,cand_buff[size]);
         size++;
         //--- проверим массив на переполненность
         if(size==default_size)
           {
            //--- увеличим размерность массива
            default_size+=100;
            ArrayResize(cand_buff,default_size);
           }
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- закроем файл
      FileClose(file_handle);
      PrintFormat("Данные прочитаны, файл %s закрыт",InpFileName);
     }
   else
     {
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
      return(INIT_FAILED);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- пустое значение
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
//--- цикл для еще необработанных свечек
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- по умолчанию 0
      open_buff[i]=0;
      close_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      //--- проверка, есть ли еще данные
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- если даты совпадают, то используем значение из файла
            if(time[i]==cand_buff[j].date)
```

© 2000-2025, MetaQuotes Ltd.

```
               {
                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;
                //--- увеличиваем счетчик
                ind=j+1;
                break;
               }
            }
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Структуры и классы, FileWriteStruct

© 2000-2025, MetaQuotes Ltd.

FileSeek

Перемещает положение файлового указателя на указанное количество байт относительно указанного положения.

```
bool  FileSeek(
   int                  file_handle,     // handle файла
   long                 offset,          // в байтах
   ENUM_FILE_POSITION   origin           // позиция для отсчета
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

offset

[in] Смещение в байтах (может принимать и отрицательное значение).

origin

[in] Точка отсчета для смещения. Может принимать одно из значений перечисления ENUM_FILE_POSITION.

Возвращаемое значение

В случае удачи функция возвращает true, в случае ошибки - false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Если результатом выполнения функции FileSeek() является отрицательное смещение (выход за "левую границу" файла), то файловый указатель будет установлен на начало файла.

Если выставить позицию за "правую границу" файла (больше, чем размер файла), то последующая запись в файл будет будет произведена не с конца файла, а с выставленной позиции. При этом между предыдущим концом файла и выставленной позицией будут записаны неопределенные значения.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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"
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string InpFileName="file.txt";     // имя файла
input string InpDirectoryName="Data";    // имя директории
```

© 2000-2025, MetaQuotes Ltd.

```
input int    InpEncodingType=FILE_ANSI; // ANSI=32 или UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- установим значение переменной для генерации случайных чисел
   _RandomSeed=GetTickCount();
//--- переменные для позиций начала строк
   ulong pos[];
   int   size;
//--- сбросим значение ошибки
   ResetLastError();
//--- откроем файл
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      //--- получим позицию начала для каждой строки в файле
      GetStringPositions(file_handle,pos);
      //--- определим сколько всего строк в файле
      size=ArraySize(pos);
      if(!size)
        {
         //--- если в файле нет строк, то завершаем работу
         PrintFormat("Файл %s пуст!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- выберем случайно номер строки
      int ind=MathRand()%size;
      //--- сдвинем позицию на начало этой строки
      if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
      //--- прочитаем и распечатаем строку с номером ind
         PrintFormat("Текст строки с номером %d: \"%s\"",ind,FileReadString(file_handl
        }
      //--- закроем файл
      FileClose(file_handle);
      PrintFormat("Файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
  }
//+------------------------------------------------------------------+
//| Функция определяет позиции начала для каждой из строк в файле и  |
//| помещает их в массив arr                                         |
//+------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- размер массива по умолчанию
   int def_size=127;
//--- выделим память для массива
   ArrayResize(arr,def_size);
//--- счетчик строк
   int i=0;
//--- если не конец файла, то есть хотя бы одна строка
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // файл пуст, выходим
//--- определим сдвиг в байтах в зависимости от кодировки
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- в цикле перебираем строки
   while(1)
     {
      //--- читаем строку
      FileReadString(handle);
      //--- проверка на конец файла
      if(!FileIsEnding(handle))
        {
         //--- запомним позицию следующей строки
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- увеличим размер массива, если он переполнен
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // конец файла, выходим
     }
//--- установим истинный размер массива
   ArrayResize(arr,i);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileSize

Возвращает размер файла в байтах.

```
ulong  FileSize(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Значение типа int.

Примечание

Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input ulong  InpThresholdSize=20;         // граница размера файлов в килобайтах
input string InpBigFolderName="big";      // папка для больших файлов
input string InpSmallFolderName="small"; // папка для маленьких файлов
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // переменная для хранения имен файлов
   string   filter="*.csv"; // фильтр для поиска файлов
   ulong    file_size=0;    // размер файла в байтах
   int      size=0;         // количество файлов
//--- распечатаем путь к папке в которой будем работать
   PrintFormat("Работаем в папке %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_P
//--- получение хэндла поиска в корне общей папки всех терминалов
   long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
//--- проверим, успешно ли отработала функция FileFindFirst()
   if(search_handle!=INVALID_HANDLE)
     {
      //--- в цикле перемещаем файлы в зависимости от их размера
      do
        {
         //--- откроем файл
         ResetLastError();
         int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
         if(file_handle!=INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             //--- получим размер файла
             file_size=FileSize(file_handle);
             //--- закроем файл
             FileClose(file_handle);
            }
          else
            {
             PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",file_name,GetLas
             continue;
            }
          //--- распечатаем размер файла
          PrintFormat("Размер файла %s равен %d байт",file_name,file_size);
          //--- определим путь для перемещения файла
          string path;
          if(file_size>InpThresholdSize*1024)
             path=InpBigFolderName+"//"+file_name;
          else
             path=InpSmallFolderName+"//"+file_name;
          //--- переместим файл
          ResetLastError();
          if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
             PrintFormat("Файл %s перемещен",file_name);
          else
             PrintFormat("Ошибка, код = %d",GetLastError());
         }
       while(FileFindNext(search_handle,file_name));
       //--- закрываем хэндл поиска
       FileFindClose(search_handle);
      }
    else
       Print("Files not found!");
  }
```

© 2000-2025, MetaQuotes Ltd.

FileTell

Возвращает текущее положение файлового указателя соответствующего открытого файла.

```
ulong  FileTell(
   int  file_handle      // handle файла
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

Возвращаемое значение

Текущая позиция файлового описателя в байтах от начала файла.

Примечание

Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
//+------------------------------------------------------------------+
//|                                                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"
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string InpFileName="file.txt";     // имя файла
input string InpDirectoryName="Data";    // имя директории
input int    InpEncodingType=FILE_ANSI; // ANSI=32 или UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- установим значение переменной для генерации случайных чисел
   _RandomSeed=GetTickCount();
//--- переменные для позиций начала строк
   ulong pos[];
   int   size;
//--- сбросим значение ошибки
   ResetLastError();
//--- откроем файл
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      PrintFormat("Файл %s открыт для чтения",InpFileName);
      //--- получим позицию начала для каждой строки в файле
      GetStringPositions(file_handle,pos);
      //--- определим сколько всего строк в файле
      size=ArraySize(pos);
      if(!size)
        {
         //--- если в файле нет строк, то завершаем работу
         PrintFormat("Файл %s пуст!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- выберем случайно номер строки
      int ind=MathRand()%size;
      //--- сдвинем позицию на начало этой строки
      FileSeek(file_handle,pos[ind],SEEK_SET);
      //--- прочитаем и распечатаем строку с номером ind
      PrintFormat("Текст строки с номером %d: \"%s\"",ind,FileReadString(file_handle))
      //--- закроем файл
      FileClose(file_handle);
      PrintFormat("Файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
  }
//+------------------------------------------------------------------+
//| Функция определяет позиции начала для каждой из строк в файле и  |
//| помещает их в массив arr                                         |
//+------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- размер массива по умолчанию
   int def_size=127;
//--- выделим память для массива
   ArrayResize(arr,def_size);
//--- счетчик строк
   int i=0;
//--- если не конец файла, то есть хотя бы одна строка
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // файл пуст, выходим
//--- определим сдвиг в байтах в зависимости от кодировки
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
```

© 2000-2025, MetaQuotes Ltd.

```
      shift=1;
   else
      shift=2;
//--- в цикле перебираем строки
   while(1)
     {
      //--- читаем строку
      FileReadString(handle);
      //--- проверка на конец файла
      if(!FileIsEnding(handle))
        {
         //--- запомним позицию следующей строки
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- увеличим размер массива, если он переполнен
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // конец файла, выходим
     }
//--- установим истинный размер массива
   ArrayResize(arr,i);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileWrite

Запись данных в файл типа CSV или TXT, разделитель между данными вставляется автоматически, если он не равен 0. После записи в файл добавляется признак конца строки "\r\n".

```
uint  FileWrite(
   int  file_handle,      // handle файла
   ...                    // список записываемых параметров
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

...

[in] Список параметров, разделенных запятыми, для записи в файл. Количество выводимых в файл параметров не должно превышать 63.

Возвращаемое значение

Количество записанных байт.

Примечание

При выводе числовые данные преобразуются в текстовый формат (см. функцию Print()). Данные типа double выводятся с точностью до 16 десятичных цифр после точки, при этом данные могут выводиться либо в традиционном либо в научном формате - в зависимости от того, как запись будет наиболее компактна. Данные типа float выводятся с 5 десятичными цифрами после точки. Для вывода вещественных чисел с другой точностью либо в явно указанном формате необходимо использовать функцию DoubleToString().

Числа типа bool выводятся в виде строк "true" или "false". Числа типа datetime выводятся в формате "YYYY.MM.DD HH:MI:SS".

Пример:

```
//+------------------------------------------------------------------+
//|                                                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"
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- параметры для получения данных из терминала
input string             InpSymbolName="EURUSD";           // валютная пара
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // таймфрейм
input int                InpFastEMAPeriod=12;              // период быстрой средней
input int                InpSlowEMAPeriod=26;              // период медленной средней
input int                InpSignalPeriod=9;                // период усреднения разно
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;       // тип цены
input datetime           InpDateStart=D'2012.01.01 00:00'; // дата начала копирования
//--- параметры для записи данных в файл
input string             InpFileName="MACD.csv";  // имя файла
input string             InpDirectoryName="Data"; // имя директории
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // дата конца копирования данных
   bool     sign_buff[]; // массив сигналов (true - покупка, false - продажа)
   datetime time_buff[]; // массив времени наступления сигналов
   int      sign_size=0; // размер массивов сигналов
   double   macd_buff[]; // массив значений индикатора
   datetime date_buff[]; // массив дат индикатора
   int      macd_size=0; // размер массивов индикатора
//--- время окончания - текущее
   date_finish=TimeCurrent();
//--- получим хэндл индикатора MACD
   ResetLastError();
   int macd_handle=iMACD(InpSymbolName,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeri
   if(macd_handle==INVALID_HANDLE)
     {
      //--- не удалось получить хэндл индикатора
      PrintFormat("Ошибка получения хэндла индикатора. Код ошибки = %d",GetLastError()
      return;
     }
//--- находимся в цикле, пока индикатор не рассчитает все свои значения
   while(BarsCalculated(macd_handle)==-1)
      Sleep(10); // задержка, чтобы индикатор успел вычислить свои значения
//--- скопируем значения индикатора за определенный период
   ResetLastError();
   if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения индикатора. Код ошибки = %d",GetLas
      return;
     }
//--- скопируем соответствующее время для значений индикатора
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения времени. Код ошибки = %d",GetLastEr
      return;
     }
//--- освободим память, занимаемую индикатором
   IndicatorRelease(macd_handle);
//--- получим размер буфера
   macd_size=ArraySize(macd_buff);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- проанализируем данные и сохраним сигналы индикатора в массивы
   ArrayResize(sign_buff,macd_size-1);
   ArrayResize(time_buff,macd_size-1);
   for(int i=1;i<macd_size;i++)
     {
      //--- сигнал на покупку
      if(macd_buff[i-1]<0 && macd_buff[i]>=0)
        {
         sign_buff[sign_size]=true;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
      //--- сигнал на продажу
      if(macd_buff[i-1]>0 && macd_buff[i]<=0)
        {
         sign_buff[sign_size]=false;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
     }
//--- откроем файл для записи значений индикатора (если его нет, то создастся автомати
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для записи",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- сначала запишем количество сигналов
      FileWrite(file_handle,sign_size);
      //--- запишем время сигналов и их значения в файл
      for(int i=0;i<sign_size;i++)
         FileWrite(file_handle,time_buff[i],sign_buff[i]);
      //--- закрываем файл
      FileClose(file_handle);
      PrintFormat("Данные записаны, файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
  }
```

Смотри также

Comment, Print, StringFormat

© 2000-2025, MetaQuotes Ltd.

FileWriteArray

Записывает в бинарный файл массивы любых типов, кроме строковых (может быть массив структур, не содержащих строки и динамические массивы).

```
uint  FileWriteArray(
   int           file_handle,          // handle файла
   const void&   array[],              // массив
   int           start=0,              // начальный индекс в массиве
   int           count=WHOLE_ARRAY     // количество элементов
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

array[]

[out] Массив для записи.

start=0

[in] Начальный индекс в массиве (номер первого записываемого элемента).

count=WHOLE_ARRAY

[in] Количество записываемых элементов (WHOLE_ARRAY означает, что записываются все элементы начиная с номера start до конца массива).

Возвращаемое значение

Количество записанных элементов.

Примечание

Строковый массив может записываться только в файл типа TXT. В этом случае строки автоматически завершаются символами конца строки "\r\n". В зависимости от типа файла ANSI или UNICODE, строки преобразовываются к ansi-кодировке, или нет.

Пример:

```
//+------------------------------------------------------------------+
//|                                          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"
//--- входные параметры
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Структура для хранения данных о ценах                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
struct prices
  {
   datetime          date; // дата
   double            bid;  // цена бид
   double            ask;  // цена аск
  };
//--- глобальные переменные
int    count=0;
int    size=20;
string path=InpDirectoryName+"//"+InpFileName;
prices arr[];
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- выделение памяти для массива
   ArrayResize(arr,size);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- запись оставшихся count строк, если count<n
   WriteData(count);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- сохраним данные в массив
   arr[count].date=TimeCurrent();
   arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- отобразим текущие данные
   Print("Date = ",arr[count].date," Bid = ",arr[count].bid," Ask = ",arr[count].ask);
//--- увеличим счетчик
   count++;
//--- если массив заполнился, то записываем данные в файл и обнуляем его
   if(count==size)
     {
      WriteData(size);
      count=0;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Запись n элементов массива в файл                                |
//+------------------------------------------------------------------+
void WriteData(const int n)
  {
//--- откроем файл
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);
   if(handle!=INVALID_HANDLE)
     {
      //--- запишем данные массива в конец файла
      FileSeek(handle,0,SEEK_END);
      FileWriteArray(handle,arr,0,n);
      //--- закрываем файл
      FileClose(handle);
     }
   else
      Print("Failed to open the file, error ",GetLastError());
  }
```

Смотри также

Переменные, FileSeek

© 2000-2025, MetaQuotes Ltd.

FileWriteDouble

Записывает в бинарный файл значение параметра типа double с текущего положения файлового указателя.

```
uint  FileWriteDouble(
   int     file_handle,     // handle файла
   double  value            // значения для записи
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

value

[in] Значение типа double.

Возвращаемое значение

В случае удачи функция возвращает количество записанных байт (в данном случае sizeof(double) =8). Файловый указатель перемещается на это же количество байт.

Пример:

```
//+------------------------------------------------------------------+
//|                                        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"
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- параметры для получения данных из терминала
input string              InpSymbolName="EURJPY";           // валютная пара
input ENUM_TIMEFRAMES     InpSymbolPeriod=PERIOD_M15;       // таймфрейм
input int                 InpMAPeriod=10;                   // период сглаживания
input int                 InpMAShift=0;                     // смещение индикатора
input ENUM_MA_METHOD      InpMAMethod=MODE_SMA;             // тип сглаживания
input ENUM_APPLIED_PRICE  InpAppliedPrice=PRICE_CLOSE;      // тип цены
input datetime            InpDateStart=D'2013.01.01 00:00'; // дата начала копирования
//--- параметры для записи данных в файл
input string              InpFileName="MA.csv";     // имя файла
input string              InpDirectoryName="Data"; // имя директории
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime date_finish=TimeCurrent();
   double   ma_buff[];
   datetime time_buff[];
   int      size;
//--- получим хэндл индикатора MA
   ResetLastError();
   int ma_handle=iMA(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMAShift,InpMAMethod,
   if(ma_handle==INVALID_HANDLE)
     {
      //--- не удалось получить хэндл индикатора
      PrintFormat("Ошибка получения хэндла индикатора. Код ошибки = %d",GetLastError()
      return;
     }
//--- находимся в цикле, пока индикатор не рассчитает все свои значения
   while(BarsCalculated(ma_handle)==-1)
      Sleep(20); // задержка, чтобы индикатор успел вычислить свои значения
   PrintFormat("В файл будут записаны значения индикатора, начиная с %s",TimeToString
//--- скопируем значения индикатора
   ResetLastError();
   if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения индикатора. Код ошибки = %d",GetLas
      return;
     }
//--- скопируем время появления соответствующих баров
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения времени. Код ошибки = %d",GetLastEr
      return;
     }
//--- получим размер буфера
   size=ArraySize(ma_buff);
//--- освободим память, занимаемую индикатором
   IndicatorRelease(ma_handle);
//--- откроем файл для записи значений индикатора (если его нет, то создастся автомати
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для записи",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- сначала запишем размер выборки данных
      FileWriteDouble(file_handle,(double)size);
      //--- запишем время и значения индикатора в файл
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteDouble(file_handle,ma_buff[i]);
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      //--- закрываем файл
      FileClose(file_handle);
      PrintFormat("Данные записаны, файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastErr
  }
```

Смотри также

Вещественные типы (double, float)

© 2000-2025, MetaQuotes Ltd.

FileWriteFloat

Записывает в бинарный файл значение параметра типа float с текущего положения файлового указателя.

```
uint  FileWriteFloat(
   int    file_handle,     // handle файла
   float  value            // записываемое значение
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

value

[in] Значение типа float.

Возвращаемое значение

В случае удачи функция возвращает количество записанных байт (в данном случае sizeof(float) =4). Файловый указатель перемещается на это же количество байт.

Пример:

```
//+------------------------------------------------------------------+
//|                                          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"
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- параметры для получения данных из терминала
input string          InpSymbolName="EURUSD";           // валютная пара
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;       // таймфрейм
input datetime        InpDateStart=D'2013.01.01 00:00'; // дата начала копирования дан
//--- параметры для записи данных в файл
input string          InpFileName="Close.bin"; // имя файла
input string          InpDirectoryName="Data"; // имя директории
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- сбросим значение ошибки
   ResetLastError();
//--- скопируем цену закрытия для каждого бара
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Не удалось скопировать значения цен закрытия. Код ошибки = %d",GetL
      return;
     }
//--- скопируем время для каждого бара
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения времени. Код ошибки = %d",GetLastEr
      return;
     }
//--- получим размер буфера
   size=ArraySize(close_buff);
//--- откроем файл для записи значений (если его нет, то создастся автоматически)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для записи",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- запишем время и значения цен закрытия в файл
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteFloat(file_handle,(float)close_buff[i]);
        }
      //--- закрываем файл
      FileClose(file_handle);
      PrintFormat("Данные записаны, файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastErr
  }
```

Смотри также

Вещественные типы (double, float), FileWriteDouble

© 2000-2025, MetaQuotes Ltd.

FileWriteInteger

Записывает в бинарный файл значение параметра типа int с текущего положения файлового указателя.

```
uint  FileWriteInteger(
   int  file_handle,         // handle файла
   int  value,               // записываемое значение
   int  size=INT_VALUE       // размер в байтах
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

value

[in] Целое значение.

size=INT_VALUE

[in] Количество байт (до 4 включительно), которые нужно записать. Предусмотрены соответствующие константы: CHAR_VALUE=1, SHORT_VALUE=2 и INT_VALUE=4, таким образом функция может записать целое значение типа char, uchar, short, ushort, int или uint

Возвращаемое значение

В случае удачи функция возвращает количество записанных байт. Файловый указатель перемещается на это же количество байт.

Пример:

```
//+------------------------------------------------------------------+
//|                                        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"
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- параметры для получения данных из терминала
input string              InpSymbolName="EURUSD";           // валютная пара
input ENUM_TIMEFRAMES     InpSymbolPeriod=PERIOD_H1;        // таймфрейм
input datetime            InpDateStart=D'2013.01.01 00:00'; // дата начала копирования
//--- параметры для записи данных в файл
input string              InpFileName="Trend.bin"; // имя файла
input string              InpDirectoryName="Data"; // имя директории
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- сбросим значение ошибки
   ResetLastError();
//--- скопируем цену закрытия для каждого бара
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Не удалось скопировать значения цен закрытия. Код ошибки = %d",GetL
      return;
     }
//--- скопируем время для каждого бара
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения времени. Код ошибки = %d",GetLastEr
      return;
     }
//--- получим размер буфера
   size=ArraySize(close_buff);
//--- откроем файл для записи значений (если его нет, то создастся автоматически)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для записи",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //---
      int   up_down=0; // флаг тенденции
      int   arr_size;  // размера массива arr
      uchar arr[];     // массив типа uchar
      //--- запишем значения времени в файл
      for(int i=0;i<size-1;i++)
        {
         //--- сравним цены закрытия на текущем и следующем барах
         if(close_buff[i]<=close_buff[i+1])
           {
            if(up_down!=1)
              {
               //--- запишем значение даты в файл, используя FileWriteInteger
               StringToCharArray(TimeToString(time_buff[i]),arr);
               arr_size=ArraySize(arr);
               //--- сначала запишем количество символов в массиве
               FileWriteInteger(file_handle,arr_size,INT_VALUE);
               //--- запишем сами символы
               for(int j=0;j<arr_size;j++)
                  FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
               //--- изменим флаг тенденции
```

© 2000-2025, MetaQuotes Ltd.

```
                up_down=1;
               }
            }
         else
            {
             if(up_down!=-1)
               {
                //--- запишем значение даты в файл, используя FileWriteInteger
                StringToCharArray(TimeToString(time_buff[i]),arr);
                arr_size=ArraySize(arr);
                //--- сначала запишем количество символов в массиве
                FileWriteInteger(file_handle,arr_size,INT_VALUE);
                //--- запишем сами символы
                for(int j=0;j<arr_size;j++)
                   FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
                //--- изменим флаг тенденции
                up_down=-1;
               }
            }
         }
      //--- закрываем файл
      FileClose(file_handle);
      PrintFormat("Данные записаны, файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
  }
```

Смотри также

IntegerToString, StringToInteger, Целые типы

© 2000-2025, MetaQuotes Ltd.

FileWriteLong

Записывает в бинарный файл значение параметра типа long с текущего положения файлового указателя.

```
uint  FileWriteLong(
   int   file_handle,     // handle файла
   long  value            // записываемое значение
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

value

[in] Значение типа long.

Возвращаемое значение

В случае удачи функция возвращает количество записанных байт (в данном случае sizeof(long) =8). Файловый указатель перемещается на это же количество байт.

Пример:

```
//+------------------------------------------------------------------+
//|                                          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"
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- параметры для получения данных из терминала
input string          InpSymbolName="EURUSD";           // валютная пара
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // таймфрейм
input datetime        InpDateStart=D'2013.01.01 00:00'; // дата начала копирования дан
//--- параметры для записи данных в файл
input string          InpFileName="Volume.bin"; // имя файла
input string          InpDirectoryName="Data";  // имя директории
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   long     volume_buff[];
   datetime time_buff[];
   int      size;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- сбросим значение ошибки
   ResetLastError();
//--- скопируем тиковые объемы для каждого бара
   if(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_bu
     {
      PrintFormat("Не удалось скопировать значения тикового объема. Код ошибки = %d",
      return;
     }
//--- скопируем время для каждого бара
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения времени. Код ошибки = %d",GetLastEr
      return;
     }
//--- получим размер буфера
   size=ArraySize(volume_buff);
//--- откроем файл для записи значений индикатора (если его нет, то создастся автомати
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для записи",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- сначала запишем размер выборки данных
      FileWriteLong(file_handle,(long)size);
      //--- запишем время и значения объема в файл
      for(int i=0;i<size;i++)
        {
         FileWriteLong(file_handle,(long)time_buff[i]);
         FileWriteLong(file_handle,volume_buff[i]);
        }
      //--- закрываем файл
      FileClose(file_handle);
      PrintFormat("Данные записаны, файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastErr
  }
```

Смотри также

Целые типы, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileWriteString

Записывает в файл типа BIN, CSV или TXT значение параметра типа string с текущего положения файлового указателя. При записи в файл типа CSV или TXT, если в строке присутствует символ '\n' (LF) без предшествующего символа '\r' (CR), то перед символом '\n' дописывается отсутствующий символ '\r'.

```
uint  FileWriteString(
   int           file_handle,     // handle файла
   const string  text_string,     // записываемая строка
   int           length=-1        // количество символов
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

text_string

[in] Строка.

length=-1

[in] Количество символов, которые нужно записать. Параметр необходим для записи строки в файл типа BIN. Если размер не указан, то записывается вся строка без завершающего 0. Если указан размер меньший, чем длина строки, то записывается часть строки без завершающего 0. Если указан размер больший, чем длина строки, то строка дописывается соответствующим количеством нулей. Для файлов типа CSV и TXT этот параметр игнорируется и строка записывается полностью.

Возвращаемое значение

В случае удачи функция возвращает количество записанных байт. Файловый указатель перемещается на это же количество байт.

Примечание

Следует иметь в виду, что при записи в файл, открытый с флагом FILE_UNICODE (либо без флага FILE_ANSI), количество записанных байт будет в 2 раза больше количества записанных символов строки. При записи в файл, открытый с флагом FILE_ANSI, количество записанных байт будет совпадать с количеством записанных символов строки.

Пример:

```
//+------------------------------------------------------------------+
//|                                          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"
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
```

© 2000-2025, MetaQuotes Ltd.

```
//--- параметры для получения данных из терминала
input string             InpSymbolName="EURUSD";           // валютная пара
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // таймфрейм
input int                InpMAPeriod=14;                   // период скользящей средне
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // тип цены
input datetime           InpDateStart=D'2013.01.01 00:00'; // дата начала копирования
//--- параметры для записи данных в файл
input string             InpFileName="RSI.csv";    // имя файла
input string             InpDirectoryName="Data"; // имя директории
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // дата конца копирования данных
   double   rsi_buff[];  // массив значений индикатора
   datetime date_buff[]; // массив дат индикатора
   int      rsi_size=0;  // размер массивов индикатора
//--- время окончания - текущее
   date_finish=TimeCurrent();
//--- получим хэндл индикатора RSI
   ResetLastError();
   int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- не удалось получить хэндл индикатора
      PrintFormat("Ошибка получения хэндла индикатора. Код ошибки = %d",GetLastError()
      return;
     }
//--- находимся в цикле, пока индикатор не рассчитает все свои значения
   while(BarsCalculated(rsi_handle)==-1)
      Sleep(10); // задержка, чтобы индикатор успел вычислить свои значения
//--- скопируем значения индикатора за определенный период
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения индикатора. Код ошибки = %d",GetLas
      return;
     }
//--- скопируем соответствующее время для значений индикатора
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения времени. Код ошибки = %d",GetLastEr
      return;
     }
//--- освободим память, занимаемую индикатором
   IndicatorRelease(rsi_handle);
//--- получим размер буфера
```

© 2000-2025, MetaQuotes Ltd.

```
   rsi_size=ArraySize(rsi_buff);
//--- откроем файл для записи значений индикатора (если его нет, то создастся автомати
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для записи",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- подготовим вспомогательные переменные
      string str="";
      bool   is_formed=false;
      //--- запишем даты формирования зон перекупленности и перепроданности
      for(int i=0;i<rsi_size;i++)
        {
         //--- проверка значений индикатора
         if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
           {
            //--- первое ли значение в этой зоне
            if(!is_formed)
              {
               //--- добавляем значение и дату
               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];
            //--- переход на следующую итерацию цикла
            continue;
           }
         //--- проверка флага
         if(is_formed)
           {
            //--- строка сформирована, запишем ее в файл
            FileWriteString(file_handle,str+"\r\n");
            is_formed=false;
           }
        }
      //--- закрываем файл
      FileClose(file_handle);
      PrintFormat("Данные записаны, файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastEr
  }
```

Смотри также

Тип string, StringFormat

© 2000-2025, MetaQuotes Ltd.

FileWriteStruct

Записывает в бинарный файл содержимое структуры, переданной в качестве параметра, с текущего положения файлового указателя.

```
uint  FileWriteStruct(
   int           file_handle,       // handle файла
   const void&   struct_object,     // ссылка на объект
   int           size=-1            // размер для записи в байтах
   );
```

Параметры

file_handle

[in] Файловый описатель, возвращаемый функцией FileOpen().

struct_object

[in] Ссылка на объект указанной структуры. Структура не должна содержать строки, динамические массивы, виртуальные функции, а также указатели на объекты и функции.

size=-1

[in] Количество байт, которые нужно записать. Если размер не указан или указано большее количество байт, чем размер структуры, то записывается вся структура полностью.

Возвращаемое значение

В случае удачи функция возвращает количество записанных байт. Файловый указатель перемещается на это же количество байт.

Пример:

```
//+------------------------------------------------------------------+
//|                                          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"
//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- параметры для получения данных из терминала
input string           InpSymbolName="EURUSD";           // валютная пара
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;         // таймфрейм
input datetime         InpDateStart=D'2013.01.01 00:00'; // дата начала копирования дан
//--- параметры для записи данных в файл
input string           InpFileName="EURUSD.txt";         // имя файла
input string           InpDirectoryName="Data";          // имя директории
//+------------------------------------------------------------------+
//| Структура для хранения данных свечи                              |
//+------------------------------------------------------------------+
struct candlesticks
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   double            open;  // цена открытия
   double            close; // цена закрытия
   double            high;  // максимальная цена
   double            low;   // минимальная цена
   datetime          date;  // дата
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
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[];
//--- сбросим значение ошибки
   ResetLastError();
//--- получим время появления баров из диапазона
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения времени. Код ошибки = %d",GetLastEr
      return;
     }
//--- получим максимальные цены баров из диапазона
   if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения максимальных цен. Код ошибки = %d",
      return;
     }
//--- получим минимальные цены баров из диапазона
   if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения минимальных цен. Код ошибки = %d",G
      return;
     }
//--- получим цены открытия баров из диапазона
   if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-1)
     {
      PrintFormat("Не удалось скопировать значения цен открытия. Код ошибки = %d",GetL
      return;
     }
//--- получим цены закрытия баров из диапазона
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Не удалось скопировать значения цен закрытия. Код ошибки = %d",GetL
      return;
     }
//--- определим размерность массивов
   size=ArraySize(time_buff);
//--- сохраним все данные в массиве структуры
   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];
     }

//--- откроем файл для записи массива структуры в файл (если его нет, то создастся авт
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Файл %s открыт для записи",InpFileName);
      PrintFormat("Путь к файлу: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_P
      //--- подготовим счетчик количества байт
      uint counter=0;
      //--- в цикле запишем значения массива
      for(int i=0;i<size;i++)
         counter+=FileWriteStruct(file_handle,cand_buff[i]);
      PrintFormat("В файл %s записано %d байт информации",InpFileName,counter);
      PrintFormat("Всего байтов: %d * %d * %d = %d, %s",size,5,8,size*5*8,size*5*8==co
      //--- закрываем файл
      FileClose(file_handle);
      PrintFormat("Данные записаны, файл %s закрыт",InpFileName);
     }
   else
      PrintFormat("Не удалось открыть файл %s, Код ошибки = %d",InpFileName,GetLastErr
  }
```

Смотри также

[Структуры и классы]

© 2000-2025, MetaQuotes Ltd.

FileLoad

Считывает всё содержимое указанного бинарного файла в переданный массив числовых типов или простых структур. Функция позволяет быстро прочитать данные известного типа в соответствующий массив.

```
long  FileLoad(
   const string  file_name,          // имя файла
   void&         buffer[],           // массив числовых типов или простых структур
   int           common_flag=0       // файловый флаг, по умолчанию файл ищется в папке
   );
```

Параметры

file_name

[in] Имя файла, из которого будет производиться считывание данных.

buffer

[out] Массив числовых типов или простых структур.

common_flag=0

[in] Файловый флаг, указывающий режим работы. Если параметр не указан, то файл ищется в подпапке MQL5\Files (или <каталог_агента_тестирования>\MQL5\Files в случае тестирования).

Возвращаемое значение

Количество прочитанных элементов или -1 в случае неудачи.

Примечание

Функция FileLoad() читает из файла количество байт, кратное размеру элемента массива. Например, пусть размер файла составляет 10 байт, а чтение производится в массив типа double (sizeof(double)=8). В этом случае будет прочитано только 8 байт, оставшиеся 2 байта с конца файла просто будут отброшены, а сама функция FileLoad() вернёт 1 (прочитан 1 элемент).

Пример:

```
//+------------------------------------------------------------------+
//|                                                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; // количество баров
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 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);
      //---  запишем котировки в файл
      if(!FileSave(filename,rates,FILE_COMMON))
         PrintFormat("FileSave() failed, error=%d",GetLastError());
     }
   else
      PrintFormat("Failed CopyRates(%s), error=",_Symbol,GetLastError());
//--- теперь прочитаем эти котировки обратно из файла
   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);
        }
     }
  }
```

Смотри также

Структуры и классы, FileReadArray, FileReadStruct, FileSave

© 2000-2025, MetaQuotes Ltd.

FileSave

Записывает в бинарный файл все элементы массива, переданного в качестве параметра. Функция позволяет быстро в одну строку записывать массивы числовых типов или простых структур.

```
bool  FileSave(
   const string  file_name,          // имя файла
   void&         buffer[],           // массив числовых типов или простых структур
   int           common_flag=0       // файловый флаг, по умолчанию файлы пишутся в пап
   );
```

Параметры

file_name

[in] Имя файла, в который будет записан массив данных.

buffer

[in] Массив числовых типов или простых структур.

common_flag=0

[in] Файловый флаг, указывающий режим работы. Если параметр не указан, то файл будет записан в подпапке MQL5\Files (или <каталог_агента_тестирования>\MQL5\Files в случае тестирования).

Возвращаемое значение

В случае неудачи функция возвращает false.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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; // количество  тиков
//+------------------------------------------------------------------+
//| 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)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat(" CopyTicks(%s) copied %d ticks",_Symbol,copied);
      //--- если тиковая история синхронизирована, то код ошибки равен нулю
      if(!GetLastError()==0)
         PrintFormat("%s: Ticks are not synchronized, error=%d",_Symbol,copied,_LastEr
      //---  запишем тики в файл
      if(!FileSave(filename,ticks,FILE_COMMON))
         PrintFormat("FileSave() failed, error=%d",GetLastError());
     }
   else
      PrintFormat("Failed CopyTicks(%s), Error=",_Symbol,GetLastError());
//--- теперь прочитаем эти тики обратно из файла
   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);
        }
     }
  }
```

Смотри также

Структуры и классы, FileWriteArray, FileWriteStruct, FileLoad, FileWrite

© 2000-2025, MetaQuotes Ltd.

FolderCreate

Создает директорию в директории Files (в зависимости от значения common_flag)

```
bool  FolderCreate(
   string  folder_name,        // строка с именем создаваемой папки
   int     common_flag=0       // область действия
   );
```

Параметры

folder_name

[in] Имя директории, которую требуется создать. Содержит относительный путь к папке.

common_flag=0

[in] Флаг, определяющий местоположение директории. Если common_flag=FILE_COMMON, то директория находится в общей папке всех клиентских терминалов \Terminal\Common\Files. В противном случае директория находится в локальной папке (MQL5\Files или MQL5\Tester\Files в случае тестирования).

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы".

Пример:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- описание
#property description "Скрипт показывает пример использования FolderCreate()."
#property description "Внешний параметр определяет папку для создания папок."
#property description "После выполнения скрипта будет создана структура папок"

//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входной параметр определяет папку, в которой работает скрипт
input bool     common_folder=false; // общая папка всех терминалов
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- папка, которую создадим в MQL5\Files
   string root_folder="Folder_A";
   if(CreateFolder(root_folder,common_folder))
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- создадим в ней дочернюю папку Child_Folder_B1
      string folder_B1="Child_Folder_B1";
      string path=root_folder+"\\"+folder_B1;          // создадим имя папки с учетом
      if(CreateFolder(path,common_folder))
        {
         //--- в этой папке создадим еще 3 дочерних
         string folder_C11="Child_Folder_C11";
         string child_path=root_folder+"\\"+folder_C11;// создадим имя папки с учетом
         CreateFolder(child_path,common_folder);
         //--- вторая дочерняя папка
         string folder_C12="Child_Folder_C12";
         child_path=root_folder+"\\"+folder_C12;
         CreateFolder(child_path,common_folder);

         //--- третья дочерняя папка
         string folder_C13="Child_Folder_C13";
         child_path=root_folder+"\\"+folder_C13;
         CreateFolder(child_path,common_folder);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Пытается создать папку и выводит сообщение об этом               |
//+------------------------------------------------------------------+
bool CreateFolder(string folder_path,bool common_flag)
  {
   int flag=common_flag?FILE_COMMON:0;
   string working_folder;
//--- выясним полный путь в зависимости от параметра common_flag
   if(common_flag)
      working_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH)+"\\MQL5\\Files";
   else
      working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- отладочное сообщение
   PrintFormat("folder_path=%s",folder_path);
//--- попытка создать папку относительно пути MQL5\Files
   if(FolderCreate(folder_path,flag))
     {
      //--- выведем полный путь для созданной папки
      PrintFormat("Создали папку %s",working_folder+"\\"+folder_path);
      //--- сбросим код ошибки
      ResetLastError();
      //--- успешное выполнение
      return true;
     }
   else
      PrintFormat("Не удалось создать папку %s. Код ошибки %d",working_folder+folder_p
```

© 2000-2025, MetaQuotes Ltd.

```
//--- неудачное выполнение
   return false;
  }
```

Смотри также

FileOpen(), FolderClean(), FileCopy()

© 2000-2025, MetaQuotes Ltd.

FolderDelete

Удаляет указанную директорию. Если папка не пуста, то она не может быть удалена.

```
bool  FolderDelete(
   string  folder_name,       // строка с именем удаляемой папки
   int     common_flag=0      // область действия
   );
```

Параметры

folder_name

[in] Имя директории, которую требуется удалить. Содержит полный путь к папке.

common_flag=0

[in] Флаг, определяющий местоположение директории. Если common_flag=FILE_COMMON, то директория находится в общей папке всех клиентских терминалов \Terminal\Common\Files. В противном случае директория находится в локальной папке (MQL5\Files или MQL5\Tester\Files в случае тестирования).

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами файловой "песочницы".

Если директория содержит хоть один файл и/или поддиректорию, то удаление такой директории невозможно, ее необходимо предварительно очистить. Тотальная очистка папки от всех файлов и всех вложенных подпапок осуществляется при помощи функции FolderClean().

Пример:

```
//+------------------------------------------------------------------+
//|                                            Demo_FolderDelete.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"
//--- описание
#property description "Скрипт показывает пример использования FolderDelete()."
#property description "Сначала создаются две папки, одна пустая, другая содержит файл.
#property description "При попытке удаления непустой папки получим ошибку и предупрежд

//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string   firstFolder="empty";     // пустая папка
```

© 2000-2025, MetaQuotes Ltd.

```
input string   secondFolder="nonempty";// папка, в которой будет один файл
string filename="delete_me.txt";       // имя файла, который мы создадим в папке secon
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- хендл файла запишем сюда
   int handle;
//--- выясним в какой папке мы работаем
   string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- отладочное сообщение
   PrintFormat("working_folder=%s",working_folder);
//--- попытка создать пустую папку относительно пути MQL5\Files
   if(FolderCreate(firstFolder,0)) // 0 означает, что работаем в локальной папке терми
     {
      //--- выведем полный путь до созданной папки
      PrintFormat("Создали папку %s",working_folder+"\\"+firstFolder);
      //--- сбросим код ошибки
      ResetLastError();
     }
   else
      PrintFormat("Не удалось создать папку %s. Код ошибки %d",working_folder+"\\"+fir

//--- теперь создадим непустую папку с помощью функции FileOpen()
   string filepath=secondFolder+"\\"+filename;  // сформируем путь для файла, который
   handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // флаг FILE_WRITE в данном случае с
   if(handle!=INVALID_HANDLE)
      PrintFormat("Открыли файл на чтение %s",working_folder+"\\"+filepath);
   else
      PrintFormat("Не удалось создать файл %s в папке %s. Код ошибки=",filename,secondo

   Comment(StringFormat("Готовимся удалить папки %s и %s", firstFolder, secondFolder))
//--- Небольшая пауза в 5 секунд, чтобы мы могли прочитать сообщение на графике
   Sleep(5000); // Sleep() нельзя использовать в индикаторах!

//--- выведем диалоговое окно и просим пользователя
   int choice=MessageBox(StringFormat("Удалить папки %s и %s?", firstFolder, secondFol
                         "Удаление папок",
                         MB_YESNO|MB_ICONQUESTION); //  будут две кнопки - "Yes" и "No

//--- выполним действия в зависимости от выбранного варианта
   if(choice==IDYES)
     {
      //--- очистим комментарий на графике
      Comment("");
      //--- выведем сообщение в журнал "Эксперты"
      PrintFormat("Пробуем удалить папки %s и %s",firstFolder, secondFolder);
      ResetLastError();
      //--- удаляем пустую папку
```

© 2000-2025, MetaQuotes Ltd.

```
      if(FolderDelete(firstFolder))
         //--- должны увидеть это сообщение, так как папка пустая
         PrintFormat("Папка %s успешно удалена",firstFolder);
      else
         PrintFormat("Не удалось удалить папку %s. Код ошибки=%d", firstFolder, GetLas

      ResetLastError();
      //--- удаляем папку, которая содержит файл
      if(FolderDelete(secondFolder))
         PrintFormat("Папка %s успешно удалена", secondFolder);
      else
         //--- должны увидеть это сообщение, так как в папке есть файл
         PrintFormat("Не удалось удалить папку %s. Код ошибки=%d", secondFolder, GetLa
     }
   else
      Print("Удаление отменено");
//---
  }
```

Смотри также

FileOpen(), FolderClean(), FileMove()

© 2000-2025, MetaQuotes Ltd.

FolderClean

Удаляет все файлы в указанной папке.

```
bool  FolderClean(
   string  folder_name,        // строка с именем папки
   int     common_flag=0       // область действия
   );
```

Параметры

folder_name

[in] Имя директории, в которой требуется удалить все файлы. Содержит полный путь к папке.

common_flag=0

[in] Флаг, определяющий местоположение директории. Если common_flag=FILE_COMMON, то директория находится в общей папке всех клиентских терминалов \Terminal\Common\Files. В противном случае директория находится в локальной папке (MQL5\Files или MQL5\Tester\Files в случае тестирования).

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Из соображений безопасности в языке MQL5 строго контролируется работа с файлами. Файлы, с которыми проводятся файловые операции средствами языка MQL5, не могут находиться за пределами "файловой песочницы".

Осторожно пользуйтесь этой функцией, так как все файлы и все вложенные поддиректории удаляются безвозвратно.

Пример:

```
//+------------------------------------------------------------------+
//|                                              Demo_FolderClean.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"
//--- описание
#property description "Скрипт показывает пример использования FolderClean()."
#property description "Сначала создаются файлы в указанной папке с помощью функции Fil
#property description "Затем перед удалением файлов выводится предупреждение MessageBo

//--- покажем окно входных параметров при запуске скрипта
#property script_show_inputs
//--- входные параметры
input string   foldername="demo_folder";  // создаем папку в MQL5/Files/
input int      files=5;                   // сколько файлов создадим и удалим
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string name="testfile";
//--- сначала откроем или создадим файлы в папке данных нашего терминала
   for(int N=0;N<files;N++)
     {
      //--- соберем имя файла в виде 'demo_folder\testfileN.txt'
      string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
      //--- открываем файл с флагом на запись, в этом случае папка 'demo_folder' будет
      int handle=FileOpen(filemane,FILE_WRITE);
      //--- выясним, насколько успешно отработала функция FileOpen()
      if(handle==INVALID_HANDLE)
        {
         PrintFormat("Не удалось создать файл %s. Код ошибки ",filemane,GetLastError()
         ResetLastError();
        }
      else
        {
         PrintFormat("Файл %s успешно открыт",filemane);
         //--- открытый файл нам больше не нужен, обязательно закрываем
         FileClose(handle);
        }
     }

//--- проверим, сколько файлов в папке
   int k=FilesInFolder(foldername+"\\*.*",0);
   PrintFormat("Всего в папке %s найдено %d файлов",foldername,k);

//--- выведем диалоговое окно и спросим пользователя
   int choice=MessageBox(StringFormat("Вы собираетесь удалить из папки %s %d файлов, п
                         "Удаление файлов из папки",
                         MB_YESNO|MB_ICONQUESTION); //  будут две кнопки - "Yes" и "No
   ResetLastError();

//--- выполним действия в зависимости от выбранного варианта
   if(choice==IDYES)
     {
      //--- начинаем удалять
      PrintFormat("Попытка удалить все файлы из папки %s",foldername);
      if(FolderClean(foldername,0))
         PrintFormat("Файлы успешно удалены, в папке %s осталось %d файлов",
                     foldername,
                     FilesInFolder(foldername+"\\*.*",0));
      else
         PrintFormat("Не удалось удалить файлы из папки %s. Код ошибки %d",foldername,
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Удаление отменено");
//---
  }
//+------------------------------------------------------------------+
//| возвращает количество файлов в указанной папке                   |
//+------------------------------------------------------------------+
int FilesInFolder(string path,int flag)
  {
   int count=0;
   long handle;
   string filename;
//---
   handle=FileFindFirst(path,filename,flag);
//--- если хотя бы один файл найден, ищем остальные
   if(handle!=INVALID_HANDLE)
     {
      //--- выведем имя файла
      PrintFormat("найден файл %s",filename);
      //--- увеличим счетчик найденный файлов/папок
      count++;
      //--- начинаем перебор всех файлов/папок
      while(FileFindNext(handle,filename))
        {
         PrintFormat("найден файл %s",filename);
         count++;
        }
      //--- обязательно закрываем хендл поиска по окончании
      FileFindClose(handle);
     }
   else // хендл получить не удалось
     {
      PrintFormat("Не удалось провести поиск файлов в папке %s",path);
     }
//--- вернем результат
   return count;
  }
```

Смотри также

FileFindFirst, FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

Пользовательские индикаторы

Группа функций, используемых при оформлении пользовательских индикаторов. Данные функции нельзя использовать при написании советников и скриптов.

Функция	Действие
SetIndexBuffer	Связывает указанный индикаторный буфер с одномерным динамическим массивом типа double
IndicatorSetDouble	Задает значение свойства индикатора, имеющего тип double
IndicatorSetInteger	Задает значение свойства индикатора,имеющего тип int
IndicatorSetString	Задает значение свойства индикатора, имеющего тип string
PlotIndexSetDouble	Задает значение свойства линии индикатора, имеющего тип double
PlotIndexSetInteger	Задает значение свойства линии индикатора, имеющего тип int
PlotIndexSetString	Задает значение свойства линии индикатора, имеющего тип string
PlotIndexGetInteger	Возвращает значение свойства линии индикатора, имеющего целый тип

Свойства индикаторов можно устанавливать как с помощью директив компилятора, так и с помощью функций. Для лучшего понимания рекомендуется изучить стили индикаторов в примерах.

Все необходимые расчеты пользовательских индикаторов необходимо размещать в предопределенной функции OnCalculate(). Если используется короткая форма вызова функции OnCalculate() вида

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin, co
```

то переменная *rates_total* содержит значение общего количества элементов массива price[], переданного в качестве входного параметра для расчета значений индикатора.

Параметр *prev_calculated* - результат выполнения функции OnCalculate() на предыдущем вызове и позволяет организовать экономный алгоритм расчета значений индикатора. Например, если текущее значение rates_total=1000, а prev_calculated=999, то, возможно, нам достаточно сделать расчеты только для одного значения каждого индикаторного буфера.

Если бы информация о размере входного массива price была бы недоступна, то это привело к необходимости производить расчеты для 1000 значений каждого индикаторного буфера. При первом вызове функции OnCalculate() значение prev_calculated=0. Если массив price[] каким-либо образом изменился, то в этом случае prev_calculated также равно 0.

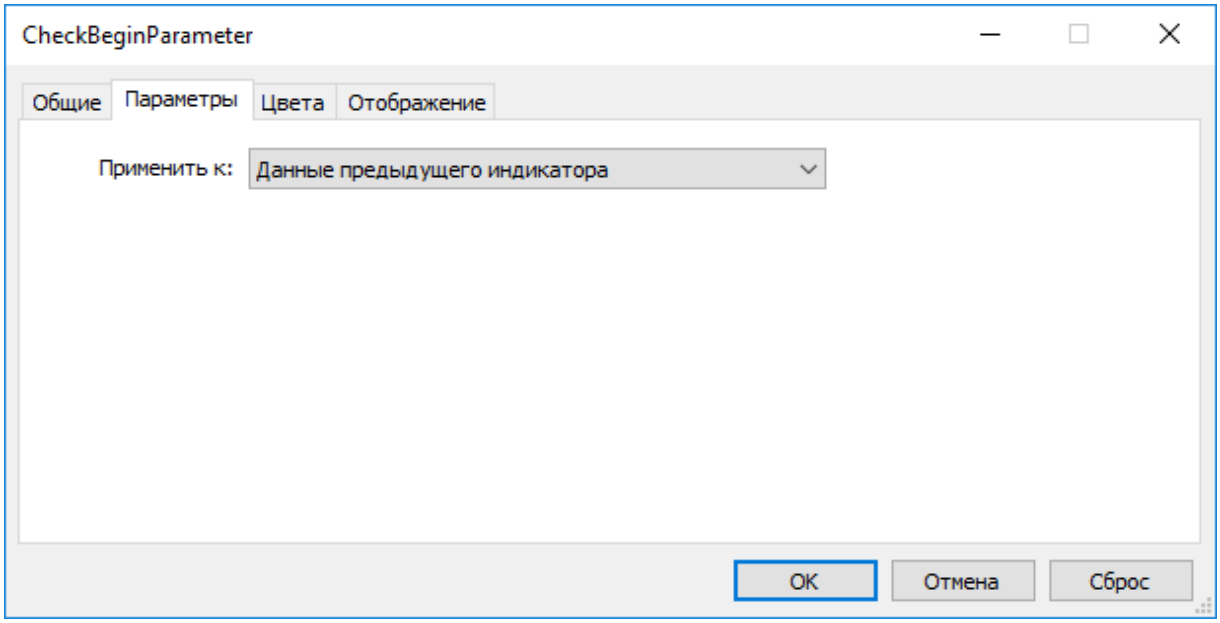
Параметр *begin* сообщает количество начальных значений массива price, которые не содержат данных для расчета. Например, если в качестве входного массива были использованы значения индикатора Accelerator Oscillator (для которого первые 37 значений не рассчитываются), то begin=37. Для примера рассмотрим простой индикатор:

© 2000-2025, MetaQuotes Ltd.

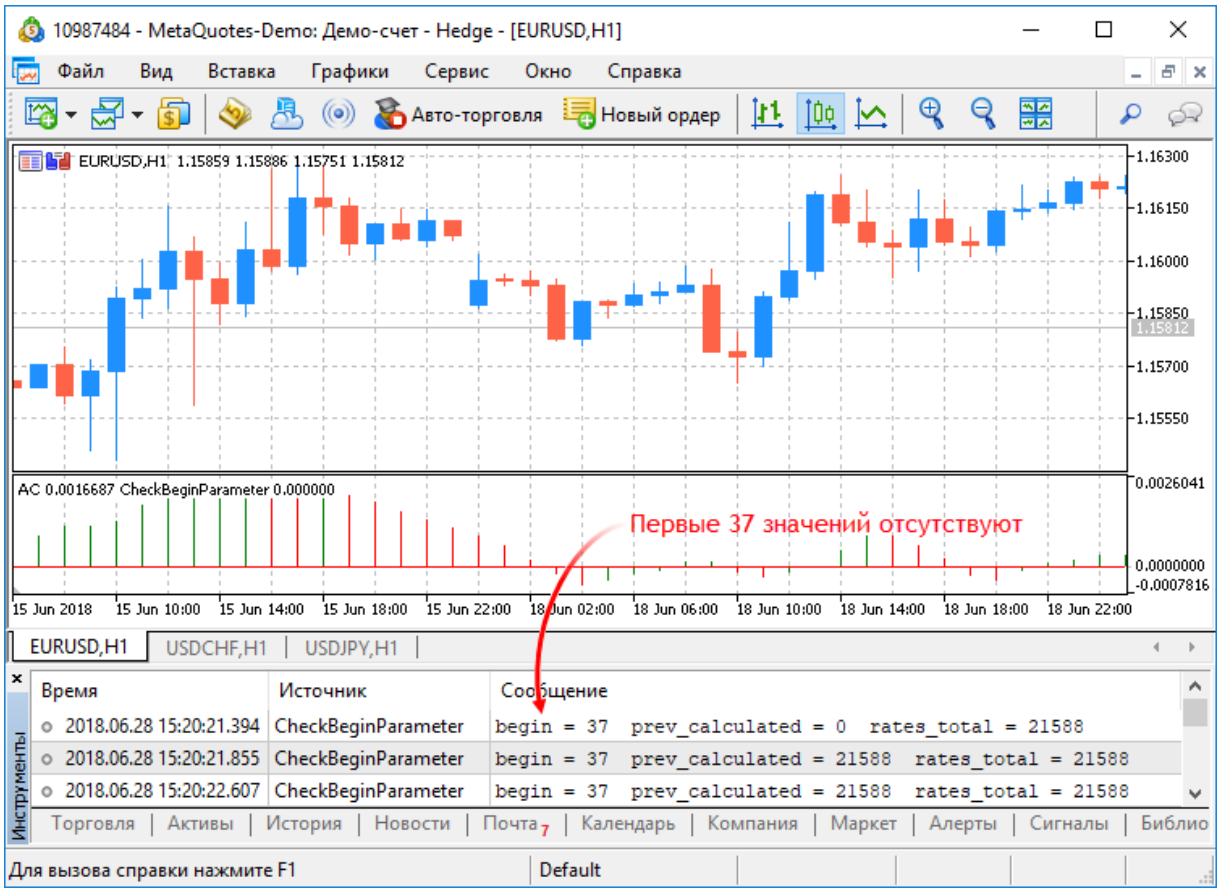
```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Перетащим его из окна "Навигатора" на окно индикатора Accelerator Oscillator и укажем, что расчеты будут производиться на значениях предыдущего индикатора:

© 2000-2025, MetaQuotes Ltd.



В результате при первом вызове функции OnCalculate() значение prev_calculated окажется равным нулю, а при последующих вызовах оно будет равно значению rates_total (до тех пор, пока не увеличится количество баров на ценовом графике).



Значение параметра begin будет в точности равно количеству начальных баров, для которых значения индикатора Accelerator не рассчитывается в соответствие с логикой этого индикатора. Если мы посмотрим исходный код пользовательского индикатора Accelerator.mq5, то увидим в функции OnInit() такие строчки:

© 2000-2025, MetaQuotes Ltd.

```
//--- sets first bar from what index will be drawn
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,37);
```

Именно функцией PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values) мы сообщаем количество несущественных первых значений в нулевом индикаторном массиве пользовательского индикатора, которые нам не нужно принимать для расчетов (empty_first_values). Таким образом, у нас есть механизмы, чтобы:

1. сообщить о количестве начальных значений индикатора, которые не стоит использовать для расчетов в другом пользовательском индикаторе;
2. получить информацию о количестве первых значений, которые необходимо игнорировать при вызове другого пользовательского индикатора, не вдаваясь в логику его расчетов.

© 2000-2025, MetaQuotes Ltd.

Стили индикаторов в примерах

В клиентский терминал MetaTrader 5 встроено 38 технических индикаторов, которые можно использовать в программах MQL5 с помощью соответствующих функций. Но главное достоинство языка MQL5 – возможность создавать свои собственные пользовательские индикаторы, которые потом можно использовать в советниках для получения значений или просто накладывать на ценовые графики для проведения технического анализа.

Все множество индикаторов можно получить на основе нескольких базовых стилей рисования, называемых графическими построениями. Под построением понимается способ отображения данных, которые индикатор рассчитывает, хранит и выдает по запросу. Всего таких базовых построений семь:

1. линия,
2. секция (отрезок),
3. гистограмма,
4. стрелка (символ),
5. закрашиваемая область (канал с заливкой),
6. бары,
7. японские свечи.

Каждое построение требует для своего отображения от одного до пяти массивов типа double, в которых хранятся значения индикатора. Эти массивы для удобства работы индикатора связываются с индикаторными буферами. Количество буферов в индикаторе необходимо объявить заранее с помощью директивы компилятора, пример:

```
#property indicator_buffers 3 // количество буферов
#property indicator_plots   2 // количество графических построений
```

Количество буферов в индикаторе всегда больше или равно количеству построений в индикаторе.

Так как каждое базовое графическое построение может иметь цветовые вариации или специфику отображения, то реальное количество построений в языке MQL5 составляет 18:

Построение	Описание	Буферов значений	Буферов цвета
DRAW_NONE	На графике визуально не отображается, но значения соответствующего буфера можно	1	-

© 2000-2025, MetaQuotes Ltd.

Построение	Описание	Буферов значений	Буферов цвета
	посмотреть в "Окне данных"		
DRAW_LINE	Строится линия по значениям соответствующего буфера (пустые значения в буфере нежелательны)	1	-
DRAW_SECTION	Рисуется отрезками между значениями соответствующего буфера (обычно пустых значений много)	1	-
DRAW_HISTOGRAM	Рисуется гистограммой от нулевой линии до значений соответ	1	-

© 2000-2025, MetaQuotes Ltd.

Построение	Описание	Буферов значений	Буферов цвета
	ствующего буфера (допускаются пустые значения)		
DRAW_HISTOGRAM2	Рисуется гистограммой на двух индикаторных буферах (допускаются пустые значения)	2	-
DRAW_ARROW	Отображается символами (допускаются пустые значения)	1	-
DRAW_ZIGZAG	Похожа на стиль DRAW_SECTION, но в отличие от него может строить вертикальные отрезки на одном баре	2	-

© 2000-2025, MetaQuotes Ltd.

Построение	Описание	Буферов значений	Буферов цвета
DRAW_FILLING	Цветовая заливка между двумя линиями. В "Окне данных" показываются 2 значения соответствующих буферов	2	-
DRAW_BARS	Отображение на графике в виде баров. В "Окне данных" показываются 4 значения соответствующих буферов	4	-
DRAW_CANDLES	Отображение в виде японских свечей. В "Окне данных" показываются 4 значения соответствующи	4	-

© 2000-2025, MetaQuotes Ltd.

Построение	Описание	Буферов значений	Буферов цвета
	х буферов		
DRAW_COLOR_LINE	Линия, для которой можно чередовать цвет как на разных барах, так и сменить ее цвет в любой момент времени	1	1
DRAW_COLOR_SECTION	Похож на стиль DRAW_SECTION, но цвет каждой секции можно задавать индивидуально, можно задавать цвет динамически	1	1
DRAW_COLOR_HISTOGRAM	Похож на стиль DRAW_HISTOGRAM, но каждая полоска может иметь свой цвет, можно	1	1

© 2000-2025, MetaQuotes Ltd.

Построение	Описание	Буферов значений	Буферов цвета
	задавать цвет динамически		
DRAW_COLOR_HISTOGRAM2	Похожа на DRAW_HISTOGRAM2, но каждая полоска может иметь свой цвет, цвет можно задавать динамически	2	1
DRAW_COLOR_ARROW	Похожа на стиль DRAW_ARROW, но каждый символ может иметь свой цвет. Цвет можно менять динамически	1	1
DRAW_COLOR_ZIGZAG	Стиль DRAW_ZIGZAG с возможностями индивидуальной раскрас	2	1

© 2000-2025, MetaQuotes Ltd.

Построение	Описание	Буферов значений	Буферов цвета
	ки секций и динамической смены цвета		
DRAW_COLOR_BARS	Стиль DRAW_BARS с возможностями индивидуальной раскраски баров и динамической смены цвета	4	1
DRAW_COLOR_CANDLES	Стиль DRAW_CANDLES с возможностями индивидуальной раскраски свечей и динамической смены цвета	4	1

Разница между индикаторным буфером и массивом

В каждом индикаторе необходимо объявить на глобальном уровне один или более массивов типа double, который затем должен быть использован в качестве индикаторного буфера с помощью функции SetIndexBuffer(). Для рисования графических построений индикатора используются только значения из индикаторных буферов, какие-либо другие массивы для этого использовать нельзя. Кроме того, значения буферов показываются в "Окне данных".

© 2000-2025, MetaQuotes Ltd.

Индикаторный буфер должен быть динамическим и не требует указания размера - размер массива, использованного в качестве индикаторного буфера, устанавливается исполняющей подсистемой терминала автоматически.

Направление индексации после связывания массива с индикаторным буфером по умолчанию устанавливается как в обычных массивах, но при необходимости можно применить функцию ArraySetAsSeries() для изменения способа доступа к элементам массива. По умолчанию индикаторный буфер используется для хранения данных, предназначенных для отрисовки (INDICATOR_DATA).

Если для расчетов значений индикатора требуется проводить промежуточные вычисления и хранить для каждого бара вспомогательное значение, то при связывании такой массив можно объявить в качестве расчетного буфера (INDICATOR_CALCULATIONS). Можно использовать для промежуточных значений и обычный массив, но в этом случае программист должен самостоятельно управлять размером такого массива.

Некоторые построения позволяют задавать для каждого бара цвет отображения. Для хранения информации о цвете используются цветовые буфера (INDICATOR_COLOR_INDEX). Цвет представлен целочисленным типом color, но все индикаторные буфера должны иметь тип double. Значения цветовых и вспомогательных (INDICATOR_CALCULATIONS) буферов нельзя получить с помощью функции CopyBuffer().

Количество индикаторных буферов должно быть указано директивой компилятора #property indicator_buffers количество_буферов:

```
#property indicator_buffers 3  //  индикатор имеет 3 буфера
```

Максимально допустимое количество буферов в одном индикаторе - 512.

Соответствие индикаторных буферов и графических построений

Каждое графическое построение базируется на одном или более индикаторных буферах. Так, для отображения простых японских свечей требуется четыре значения - цены Open, High, Low и Close. Соответственно, для отображения индикатора в виде японских свечей необходимо объявить 4 индикаторных буфера и 4 массива типа double под них. Например:

```
//--- в индикаторе четыре индикаторных буфера
#property indicator_buffers 4
//--- в индикаторе одно графическое построение
#property indicator_plots   1
//--- графическое построение под номером 1 будет отображаться японскими свечами
#property indicator_type1   DRAW_CANDLES
//--- японские свечи будут рисоваться цветом clrDodgerBlue
#property indicator_color1  clrDodgerBlue
//--- 4 массива под индикаторные буферы
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

© 2000-2025, MetaQuotes Ltd.

Графические построения автоматически используют индикаторные буферы в соответствии с номером построения. Номера построения начинаются с единицы, номера буферов начинаются с нуля. Если первое построение требует 4 индикаторных буфера, то для отрисовки будут использованы 4 первых индикаторных буфера. Эти четыре буфера должны быть связаны функцией SetIndexBuffer() с соответствующими массивами с правильной индексацией.

```
//--- связывание массивов с индикаторными буферами
   SetIndexBuffer(0,OBuffer,INDICATOR_DATA);  // первый буфер соответствует нулевому и
   SetIndexBuffer(1,HBuffer,INDICATOR_DATA);  // второй буфер соответствует индексу 1
   SetIndexBuffer(2,LBuffer,INDICATOR_DATA);  // третий буфер  соответствует индексу 2
   SetIndexBuffer(3,CBuffer,INDICATOR_DATA);  // четвертый буфер  соответствует индекс
```

При отрисовке японских свечей индикатор будет использовать именно первые четыре буфера, потому что построение "японские свечи" было объявлено под первым номером.

Изменим немного пример, добавим построение в виде простой линии - DRAW_LINE. Пусть теперь линия будет иметь номер 1, а японские свечи будут под номером 2. Количество буферов и количество построений увеличилось.

```
//--- в индикаторе 5 индикаторных буферов
#property indicator_buffers 5
//--- в индикаторе 2 графических построения
#property indicator_plots   2
//--- графическое построение под номером 1 будет отображаться линией
#property indicator_type1   DRAW_LINE
//--- линия будет рисоваться цветом clrDodgerRed
#property indicator_color1  clrDodgerRed
//--- графическое построение под номером 2 будет отображаться японскими свечами
#property indicator_type2   DRAW_CANDLES
//--- японские свечи будут рисоваться цветом clrDodgerBlue
#property indicator_color2  clrDodgerBlue
//--- 5 массивов под индикаторные буферы
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

Порядок следования построений изменился, теперь первым идет линия, следом японские свечи. Поэтому и порядок следования буферов будет таким же - сначала объявим под нулевым индексом буфер для линии, а затем четыре буфера для отображения японских свечей.

```
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);  // первый буфер соответствует индексу
//--- связывание массивов с индикаторными буферами под японские свечи
   SetIndexBuffer(1,OBuffer,INDICATOR_DATA);     // второй буфер соответствует индексу
   SetIndexBuffer(2,HBuffer,INDICATOR_DATA);     // третий буфер соответствует индексу
   SetIndexBuffer(3,LBuffer,INDICATOR_DATA);     // четвертый буфер  соответствует инд
   SetIndexBuffer(4,CBuffer,INDICATOR_DATA);     // пятый буфер  соответствует индексу
```

© 2000-2025, MetaQuotes Ltd.

Количество буферов и графических построений можно задавать только с помощью директив компилятора, динамическое изменение этих свойств с помощью функций невозможно.

Цветовые версии стилей

Как можно увидеть в таблице, стили делятся на две группы. Первая группа — это стили, у которых в названии нет слова **COLOR**, назовем эти стили базовыми:

- DRAW_LINE
- DRAW_SECTION
- DRAW_HISTOGRAM
- DRAW_HISTOGRAM2
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_FILLING
- DRAW_BARS
- DRAW_CANDLES

Вторая группа стилей содержит в названии слово **COLOR**, назовем их цветовыми версиями:

- DRAW_COLOR_LINE
- DRAW_COLOR_SECTION
- DRAW_COLOR_HISTOGRAM
- DRAW_COLOR_HISTOGRAM2
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

Все цветовые версии стилей отличаются от базовых тем, что позволяют задавать цвет для каждой части графического построения. Минимальной частью построения является бар, поэтому можно сказать, что цветовые версии позволяют задавать цвет построения на каждом баре.

Для того чтобы задавать цвет построения на каждом баре, в цветовые версии стилей был добавлен дополнительный специальный буфер для хранения индекса цвета. Эти индексы указывают на номер цвета в специальном массиве, содержащем заранее определенный набор цветов. Размер массива цветов - 64. Это означает, что каждая цветовая версия стиля позволяет раскрасить графическое построение 64-мя различными цвета.

Набор и количество цветов в специальном массиве цветов можно задавать директивой компилятора #property indicator_color, где через запятую указываются все необходимые цвета. Например, такая запись в индикаторе:

```
//--- зададим 8 цветов для раскраски свечей (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
```

© 2000-2025, MetaQuotes Ltd.

Здесь указано, что для графического построения номер 1 заданы 8 цветов, которые будут помещены в специальный массив. Далее в программе мы будем указывать не сам цвет, которым будет отображаться графическое построение, а только его индекс. Если мы хотим задать для бара красный цвет, то для этого необходимо установить в цветовом буфере индекс красного цвета из массива. Красный цвет задан в директиве первым первым, ему соответствует индекс номер 0.

```
//--- зададим цвет свечи clrRed
col_buffer[buffer_index]=0;
```

Набор цветов для раскрашивания не является раз и навсегда заданным, его можно менять динамически с помощью функции PlotIndexSetInteger(). Пример:

```
      //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                      //  номер графического стиля
                          PLOT_LINE_COLOR,        //  идентификатор свойства
                          plot_color_ind,         //  индекс цвета, куда запишем цвет
                          color_array[i]);        //  новый цвет
```

Свойства индикатора и графических построений

Для графических построений свойства можно устанавливать как с помощью директив компилятора, так и с помощью соответствующих функций. Подробней об этом написано в разделе Связь между свойствами индикатора и функциями. Динамическое изменение свойств индикатора с помощью функций позволяет создавать более гибкие пользовательские индикаторы.

Начало отрисовки индикатора на графике

Во многих случаях по условиям алгоритма расчет значений индикатора невозможно начать сразу же с текущего бара, требуется обеспечить минимальное количество предыдущих доступных баров в истории. Например, многие виды сглаживания подразумевают, что берется массив цен за предыдущие N баров, и на основании этих значений рассчитывается значение индикатора для текущего бара.

В таких случаях либо нет возможности рассчитать значения индикатора на N самых первых барах, либо эти значения не предназначены для отображения на графике и являются только вспомогательными для расчета последующих значений. Чтобы отказаться от визуализации индикатора на первых N барах истории, следует установить свойству PLOT_DRAW_BEGIN значение N для соответствующего графического построения:

```
//--- связывание массивов с индикаторными буферами под японские свечи
PlotIndexSetInteger(номер_графического_построения,PLOT_DRAW_BEGIN,N);
```

Здесь:

- номер_графического_построения - значение от нуля до indicator_plots-1 (нумерация графических построений начинается с нуля).
- N - количество первых в истории баров, на которых индикатор не должен отображаться на графике.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

DRAW_NONE

Стиль DRAW_NONE предназначен для тех случаев, когда значения буфера необходимо рассчитывать и показывать в "Окне данных", но само отображение на графике не требуется. Для настройки точности отображения используйте в функции OnInit() выражение IndicatorSetInteger(INDICATOR_DIGITS,количество_знаков):

```
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- установим точность, с которой значение будет показываться в Окне данных
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
```

Количество требуемых буферов для построения DRAW_NONE — 1.

Пример индикатора, который показывает в "Окне данных" номер бара, на котором находится мышка. Нумерация соответствует таймсерии, то есть текущий незавершенный бар имеет нулевой индекс, а самый старый бар имеет самый большой индекс.



Обратите внимание, что несмотря на то, что для графического построения №1 указан красный цвет отображения, индикатор ничего не рисует на графике.

```
//+------------------------------------------------------------------+
//|                                                   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
//--- plot Invisible
#property indicator_label1  "Bar Index"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- indicator buffers
double         InvisibleBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- связывание массива и индикаторного буфера
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- установим точность, с которой значение будет показываться в "Окне данных"
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- если это первый расчет индикатора
   if(prev_calculated==0)
     {
      //--- перенумеруем бары в первый раз
      CalcValues(rates_total,close);
      //--- запомним время открытия текущего бара в lastbar
      lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    else
      {
       //--- если появился новый бар, время его открытия не совпадает с lastbar
       if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
         {
          //--- перенумеруем бары заново
          CalcValues(rates_total,close);
          //--- обновим время открытия текущего бара в lastbar
          lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Нумерует бары как в таймсерии                                    |
//+------------------------------------------------------------------+
void CalcValues(int total,double const  &array[])
  {
//--- зададим индикаторному буферу индексацию как в таймсерии
   ArraySetAsSeries(InvisibleBuffer,true);
//--- заполним каждому бару его номер
   for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
  }
```

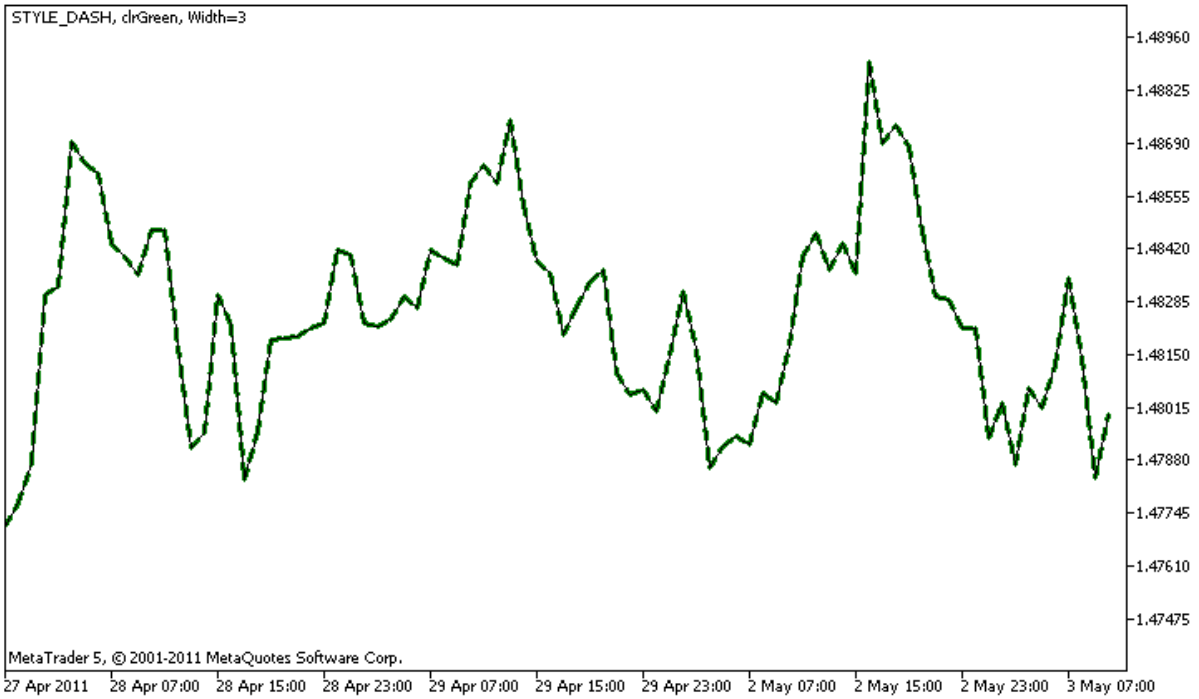
© 2000-2025, MetaQuotes Ltd.

DRAW_LINE

Стиль DRAW_LINE рисует заданным цветом линию по значениям индикаторного буфера. Толщину, цвет и стиль отображения линии можно задавать как директивами компилятора, так и динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет создавать "живые" индикаторы, которые меняют свой вид в зависимости от текущей ситуации.

Количество требуемых буферов для построения DRAW_LINE — 1.

Пример индикатора, рисующего линию ценам закрытия баров Close. Цвет, толщина и стиль линии меняются случайным образом каждые N=5 тиков.



Обратите внимание, первоначально для графического построения **plot1** со стилем DRAW_LINE свойства задаются с помощью директивы компилятора #property, а затем в функции OnCalculate() эти три свойства задаются случайным образом. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор для демонстрации DRAW_LINE"
#property description "Рисует линию заданным цветом по ценам Close"
#property description "Цвет, толщина и стиль линии меняется случайным образом"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- свойства линии заданы с помощью директив компилятора
#property indicator_label1  "Line"      // название построения для "Окна данных"
#property indicator_type1   DRAW_LINE   // тип графического построения - линия
#property indicator_color1  clrRed      // цвет линии
#property indicator_style1  STYLE_SOLID // стиль линии
#property indicator_width1  1           // толщина линии
//--- input параметр
input int      N=5;          // кол-во тиков для изменения
//--- индикаторный буфер для построения
double         LineBuffer[];
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- связывание массива и индикаторного буфера
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- инициализация генератора псевдослучайных чисел
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось критическое число тиков
```

© 2000-2025, MetaQuotes Ltd.

```
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- блок расчета значений индикатора
   for(int i=0;i<rates_total;i++)
     {
      LineBuffer[i]=close[i];
     }

//--- вернем значение prev_calculated для следующего вызова функции
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид отображаемой линии в индикаторе             |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
//--- блок изменения цвета линии
//--- получим случайное число
   int number=MathRand();
//--- делитель числа равен размеру массива colors[]
   int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
   comm=comm+(string)colors[color_index];

//--- блок изменения толщины линии
   number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5; // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+", Width="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   size=ArraySize(styles);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm=EnumToString(styles[style_index])+", "+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_SECTION

Стиль DRAW_SECTION рисует заданным цветом отрезки по значениям индикаторного буфера. Толщину, цвет и стиль отображения линии можно задавать так же, как и для стиля DRAW_LINE - директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет "оживить" индикаторы, чтобы они меняли свой вид в зависимости от текущей ситуации.

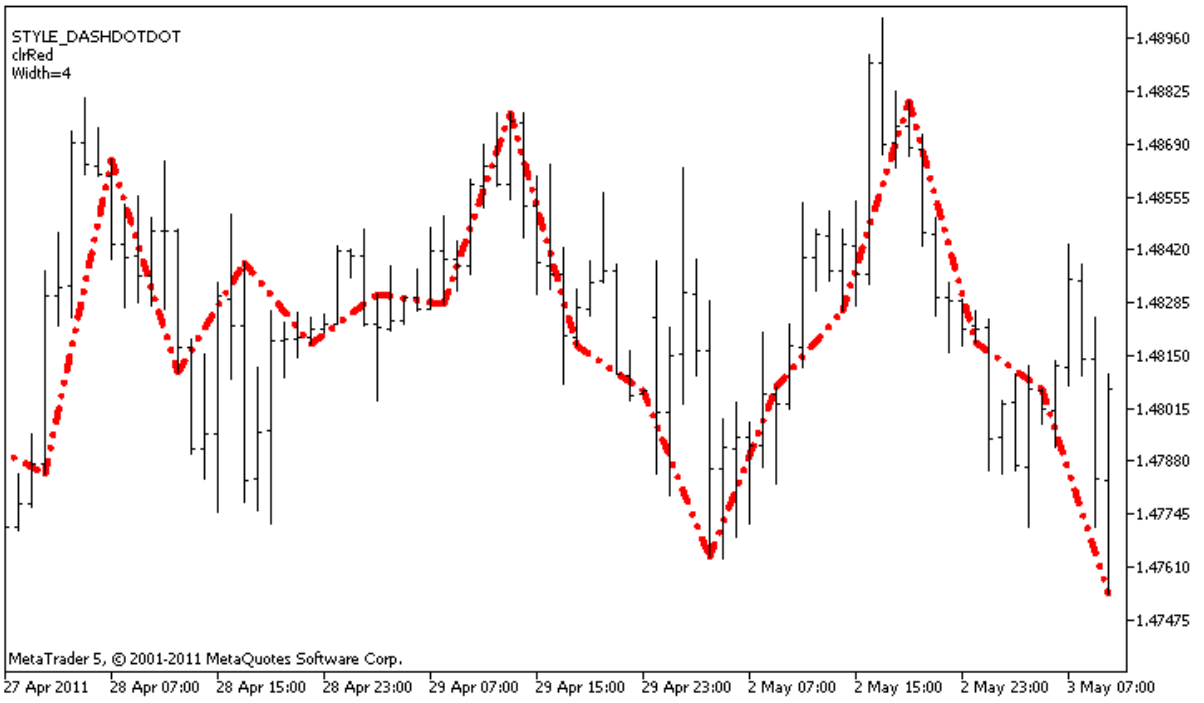
Секции рисуются от одного непустого значения до другого непустого значения индикаторного буфера, пустые значения пропускаются. Чтобы указать, какое значение следует считать "пустым", установите это значение в свойстве PLOT_EMPTY_VALUE. Например, если индикатор должен рисоваться отрезками по ненулевым значениям, то нужно задать нулевое значение в качестве пустого:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(индекс_построения_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

Всегда явно заполняйте значениями все элементы индикаторного буфера, неотрисовываемым элементам задавайте пустое значение.

Количество требуемых буферов для построения DRAW_SECTION — 1.

Пример индикатора, рисующего отрезки между ценами High и Low. Цвет, толщина и стиль **всех** секций меняются случайным образом каждые **N** тиков.



Обратите внимание, первоначально для графического построения **plot1** со стилем DRAW_SECTION свойства задаются с помощью директивы компилятора #property, а затем в функции OnCalculate() эти три свойства задаются случайным образом. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                  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 "Индикатор для демонстрации DRAW_SECTION"
#property description "Рисует прямыми отрезками через каждые bars баров"
#property description "Цвет, толщина и стиль секций меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Section
#property indicator_label1  "Section"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input параметр
input int      bars=5;           // длина секций в барах
input int      N=5;              // кол-во тиков для изменения стиля секций
//--- индикаторный буфер для построения
double         SectionBuffer[];
//--- вспомогательная переменная для вычисления концов секций
int            divider;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- связывание массива и индикаторного буфера
   SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- проверим параметр индикатора
   if(bars<=0)
     {
      PrintFormat("Недопустимое значение параметра bar=%d",bars);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=2*bars;
```

© 2000-2025, MetaQuotes Ltd.

```
//---+
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось критическое число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- номер бара, с которого начнем расчет значений индикатора
   int start=0;
//--- если индикатор уже рассчитывали раньше, то установим start на предыдущий бар
   if(prev_calculated>0) start=prev_calculated-1;
//--- здесь все расчеты значений индикатора
   for(int i=start;i<rates_total;i++)
     {
      //--- получим остаток от деления номера бара на 2*bars
      int rest=i%divider;
      //--- если номер бара делится без остатка на 2*bars
      if(rest==0)
        {
         //--- конец отрезка установим на цену High этого бара
         SectionBuffer[i]=high[i];
        }
      //--- если остаток от деления равен bars,
      else
        {
         //--- конец отрезка установим на цену High этого бара
         if(rest==bars) SectionBuffer[i]=low[i];
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- если ничего не подошло, то этот бар пропускаем - ставим значение 0
         else SectionBuffer[i]=0;
        }
     }
//--- вернем значение prev_calculated для следующего вызова функции
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид секций в индикаторе                         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
//--- блок изменения цвета линии
   int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
   int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
   comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения толщины линии
   number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5;   // толщина задается от 0 до 4
//--- установим толщину
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
//--- установим стиль линий
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

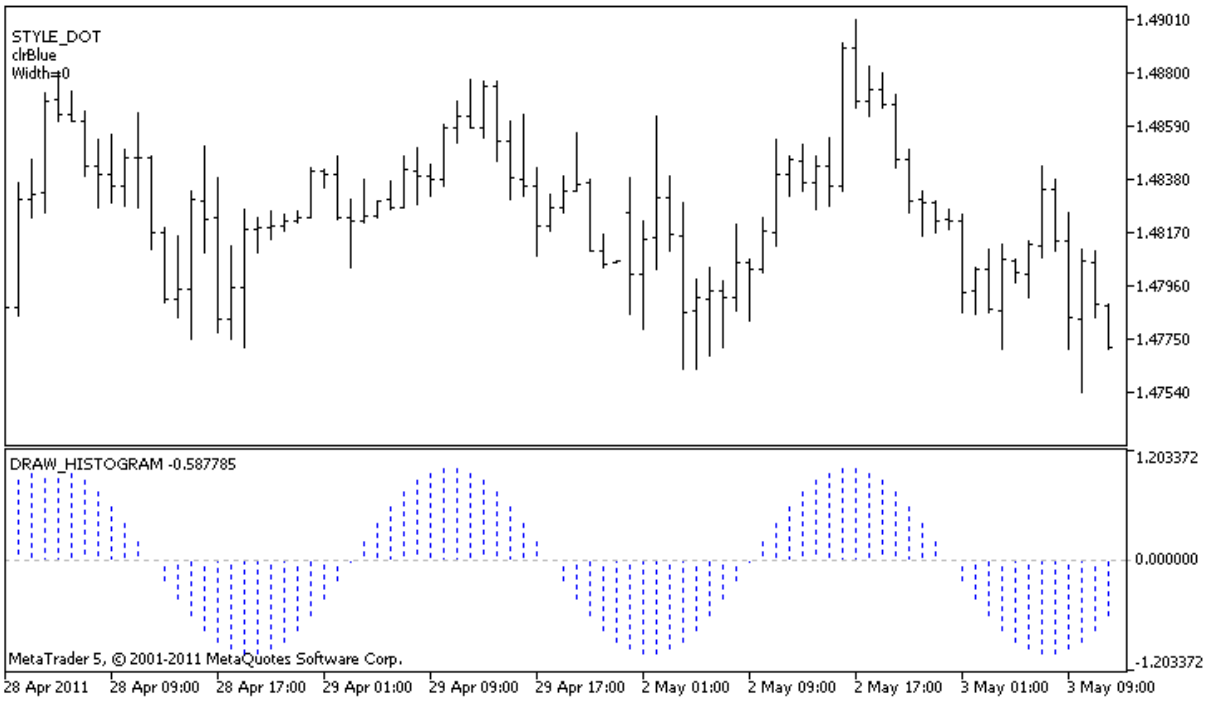
DRAW_HISTOGRAM

Стиль DRAW_HISTOGRAM рисует заданным цветом гистограмму столбиками от нуля до указанного значения. Значения берутся из индикаторного буфера. Толщину, цвет и стиль отображения стобика можно задавать так же, как и для стиля DRAW_LINE - директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет изменять вид гистограммы в зависимости от текущей ситуации.

Так как на каждом баре рисуется столбик от нулевого уровня, то DRAW_HISTOGRAM лучше использовать для отображения в отдельном подокне графика. Чаще всего этот тип графического построения используется для создания индикаторов осцилляторного типа, например, Bears Power или OsMA. Для пустых неотображаемых значений достаточно указывать нулевое значение.

Количество требуемых буферов для построения DRAW_HISTOGRAM — 1.

Пример индикатора, рисующего заданным цветом синусоиду по функции MathSin(). Цвет, толщина и стиль **всех** столбиков гистограммы меняются случайным образом каждые **N** тиков. Параметр bars задает период синусоиды, то есть через заданное количество баров синусоида будет повторяться по циклу.



Обратите внимание, что первоначально для графического построения **plot1** со стилем DRAW_HISTOGRAM свойства задаются с помощью директивы компилятора #property, а затем в функции OnCalculate() эти три свойства задаются случайным образом. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_HISTOGRAM"
#property description "Рисует синусоиду гистограммой в отдельном окне"
#property description "Цвет и толщина столбиков меняется случайным образом"
#property description "через каждые N тиков"
#property description "Параметр bars задает количество баров в цикле синусоиды"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input параметры
input int      bars=30;          // период синусоиды в барах
input int      N=5;              // кол-во тиков для изменения гистограммы
//--- indicator buffers
double         HistogramBuffer[];
//--- множитель для получения угла 2Pi в радианах при умножении на параметр bars
double    multiplier;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- вычислим множитель
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Задайте значение bars=%d больше 1",bars);
      //--- досрочное прекращение работы индикатора
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось критическое число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- вычисления значений индикатора
   int start=0;
//--- если расчет уже производился на предыдущем запуске OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // установим начало расчетов с предп
//--- заполняем индикаторный буфер значениями
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=sin(i*multiplier);
     }
//--- вернем значение prev_calculated для следующего вызова функции
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид линий в индикаторе                          |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
//--- блок изменения цвета линии
   int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
   int size=ArraySize(colors);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
   comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения толщины линии
   number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5;   // толщина задается от о до 4
//--- установим толщину
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
//--- установим стиль линий
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_HISTOGRAM2

Стиль DRAW_HISTOGRAM2 рисует заданным цветом гистограмму - вертикальные отрезки по значениям двух индикаторных буферов. Толщину, цвет и стиль отображения отрезков можно задавать так же, как и для стиля DRAW_LINE - директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет изменять вид гистограммы в зависимости от текущей ситуации.

Стиль DRAW_HISTOGRAM можно использовать как в отдельном подокне графика, так и в главном окне. Для пустых значений отрисовка не производится, все значения в индикаторных буферах нужно устанавливать явным образом. Инициализация буферов пустым значением не производится.

Количество требуемых буферов для построения DRAW_HISTOGRAM2 — 2.

Пример индикатора, рисующего на каждом баре вертикальный отрезок заданного цвета и толщины между ценами Open и Close. Цвет, толщина и стиль **всех** столбиков гистограммы меняются случайным образом каждые **N** тиков. При запуске индикатора в функции OnInit() случайным образом устанавливается номер дня недели, для которого гистограмма не будет рисоваться - invisible_day. Для этого задается пустое значение PLOT_EMPTY_VALUE=0:

```
//--- установим пустое значение
   PlotIndexSetDouble(индекс_построения_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```



Обратите внимание, что первоначально для графического построения **plot1** со стилем DRAW_HISTOGRAM2 свойства задаются с помощью директивы компилятора #property, а затем в функции OnCalculate() эти три свойства задаются случайным образом. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM2.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

© 2000-2025, 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 "Индикатор для демонстрации DRAW_HISTOGRAM2"
#property description "Рисует на каждом баре отрезок между Open и Close"
#property description "Цвет, толщина и стиль меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot 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
//--- input parameters
input int      N=5;              // кол-во тиков для изменения гистограммы
//--- indicator buffers
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- день недели, для которого индикатор не рисуется
int invisible_day;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//--- установим пустое значение
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- получим случайное число от 0 до 5
   invisible_day=MathRand()%6;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

© 2000-2025, 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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось критическое число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- вычисления значений индикатора
   int start=0;
//--- для получения дня недели по времени открытия каждого бара
   MqlDateTime dt;
//--- если расчет уже производился на предыдущем запуске OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // установим начало расчетов с предп
//--- заполняем индикаторный буфер значениями
   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];
        }
     }
//--- вернем значение prev_calculated для следующего вызова функции
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид линий в индикаторе                          |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
//--- блок изменения цвета линии
   int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
   int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
   comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения толщины линии
   number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5;   // толщина задается от 0 до 4
//--- установим толщину линий
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
//--- установим стиль линий
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- добавим информацию о дне, который пропускается в расчетах
   comm="\r\nНеотрисовываемый день - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_ARROW

Стиль DRAW_ARROW рисует на графике заданным цветом стрелки (символы из набора Wingdings) по значению индикаторного буфера. Толщину и цвет символов можно задавать так же, как и для стиля DRAW_LINE - директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет изменять вид индикатора в зависимости от текущей ситуации.

Код символа для вывода на график задается с помощью свойства PLOT_ARROW.

```
//--- зададим код символа из шрифта Wingdings для отрисовки в PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

По умолчанию значение PLOT_ARROW=159 (кружок).

Каждая стрелка фактически представляет собой символ, который имеет высоту и точку привязки, и может закрывать собою некоторую важную информацию на графике (например, цену закрытия на баре). Поэтому можно дополнительно указать вертикальное смещение в пикселях, которое не зависит от масштаба графика. На это указанное количество пикселей стрелки будут визуально смещены по вертикали, хотя сами значения индикатора останутся те же самые:

```
//--- зададим смещение стрелок по вертикали в пикселях
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

Отрицательное значение PLOT_ARROW_SHIFT означает смещение стрелок вверх, положительное значение смещает стрелки вниз.

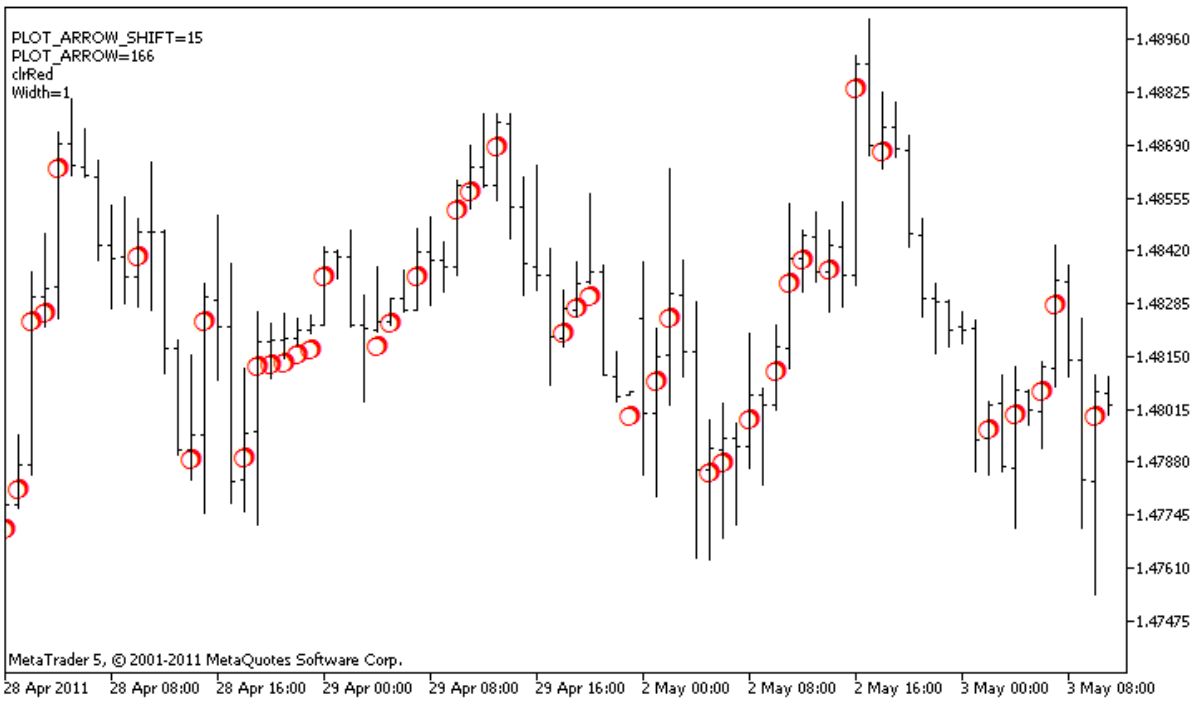
Стиль DRAW_ARROW можно использовать как в отдельном подокне графика, так и в главном окне. Пустые значения не отрисовываются и не отображаются в "Окне данных", все значения в индикаторных буферах нужно устанавливать явным образом. Инициализация буферов пустым значением не производится.

```
//--- установим пустое значение
   PlotIndexSetDouble(индекс_построения_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

Количество требуемых буферов для построения DRAW_ARROW — 1.

Пример индикатора, рисующего стрелки на каждом баре, у которого цена закрытия Close больше цены закрытия предыдущего бара. Цвет, толщина, смещение и код символа **всех** стрелок меняются случайным образом каждые **N** тиков.

© 2000-2025, MetaQuotes Ltd.



В примере первоначально для графического построения **plot1** со стилем DRAW_ARROW свойства, цвет и размер задаются с помощью директивы компилятора #property, а затем в функции OnCalculate() свойства задаются случайным образом. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_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 "Индикатор для демонстрации DRAW_ARROW"
#property description "Рисует на графике стрелки, задаваемые символами Unicode"
#property description "Цвет, размер, смещение и код символа стрелки меняется случайным
#property description "через каждые N тиков"
#property description "Параметр code задает базовое значение: код=159 (кружок)"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Arrows
#property indicator_label1  "Arrows"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- input параметры
```

© 2000-2025, MetaQuotes Ltd.

```
input int      N=5;          // кол-во тиков для изменения
input ushort   code=159;     // код символа для отрисовки в DRAW_ARROW
//--- индикаторный буфер для построения
double         ArrowsBuffer[];
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- зададим код символа для отрисовки в PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- зададим смещение стрелок по вертикали в пикселях
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- установим в качестве пустого значения 0
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения цвета, размера, смешения и кода стрелки
   ticks++;
//--- если накопилось критическое число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- блок расчета значений индикатора
```

© 2000-2025, MetaQuotes Ltd.

```
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- цикл расчета
   for(int i=1;i<rates_total;i++)
     {
      //--- если текущая цена Close больше предыдущей, ставим стрелку
      if(close[i]>close[i-1])
         ArrowsBuffer[i]=close[i];
      //--- в противном случае указываем нулевое значение
      else
         ArrowsBuffer[i]=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид символов в индикаторе                       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах индикатора
   string comm="";
//--- блок изменения цвета стрелки
   int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
   int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
   comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения размера стрелок
   number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5;   // размер задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем размер стрелок
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения кода стрелки (PLOT_ARROW)
   number=MathRand();
//--- получим остаток от целочисленного деления для вычисления нового кода стрелки (от
   int code_add=number%20;
//--- установим новый код символа как сумму code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- запишем код символа PLOT_ARROW
```

© 2000-2025, MetaQuotes Ltd.

```
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- блок изменения смещения стрелок по вертикали в пикселях
   number=MathRand();
//--- получим смещение как остаток от целочисленного деления
   int shift=20-number%41;
//--- установим новое смещение от -20 до 20
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- запишем смещение PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_ZIGZAG

Стиль DRAW_ZIGZAG рисует заданным цветом отрезки по значениям двух индикаторных буферов. Этот стиль очень похож на DRAW_SECTION, но в отличие от последнего позволяет рисовать вертикальные отрезки в пределах одного бара, если для этого бара заданы значения для обоих индикаторных буферов. Отрезки рисуются от значения в первом буфере до значения во втором индикаторном буфере. Ни один из буферов не может содержать только пустые значения, так как в этом случае отрисовка не происходит.

Толщину, цвет и стиль отображения линии можно задавать так же, как и для стиля DRAW_SECTION - директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет "оживить" индикаторы, чтобы они меняли свой вид в зависимости от текущей ситуации.

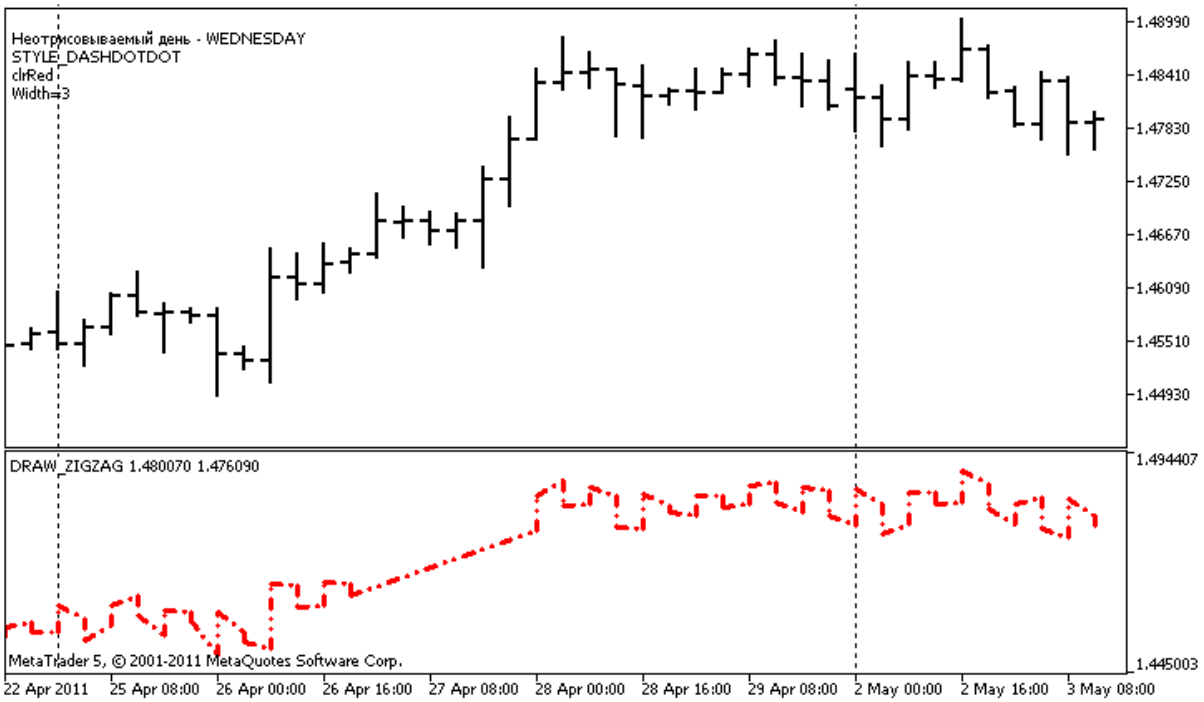
Секции рисуются от одного непустого значения одного буфера до непустого значения другого индикаторного буфера. Чтобы указать, какое значение следует считать "пустым", установите это значение в свойстве PLOT_EMPTY_VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(индекс_построения_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Всегда явно заполняйте значениями индикаторные буфера, на пропускаемых барах указывайте в буфере пустое значение.

Количество требуемых буферов для построения DRAW_ZIGZAG — 2.

Пример индикатора, рисующего пилу по ценам High и Low. Цвет, толщина и стиль линий зигзага меняются случайным образом каждые N тиков.



Обратите внимание, первоначально для графического построения **plot1** со стилем DRAW_ZIGZAG свойства задаются с помощью директивы компилятора #property, а затем в функции OnCalculate() эти три свойства задаются случайным образом. Параметр N вынесен во внешние параметры

© 2000-2025, MetaQuotes Ltd.

индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                                  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 "Индикатор для демонстрации DRAW_ZIGZAG"
#property description "Рисует прямыми отрезками \"пилу\", пропуская бары одного дня"
#property description "День пропусков выбирается случайным образом при запуске индика
#property description "Цвет, толщина и стиль отрезков меняется случайным"
#property description " образом через каждые N тиков"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ZigZag
#property indicator_label1  "ZigZag"
#property indicator_type1   DRAW_ZIGZAG
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input параметры
input int      N=5;              // кол-во тиков для изменения
//--- indicator buffers
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- день недели, для которого индикатор не рисуется
int invisible_day;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- связывание массивов и индикаторных буферов
   SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- получим случайное число от 0 до 6, для этого дня индикатор не рисуется
   invisible_day=MathRand()%6;
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось достаточное число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- структура времени понадобится для получения дня недели каждого бара
   MqlDateTime dt;

//--- позиция начала расчетов
   int start=0;
//--- если индикатор рассчитывался на предыдущем тике, то начинаем расчет с предпослед
   if(prev_calculated!=0) start=prev_calculated-1;
//--- цикл расчетов
   for(int i=start;i<rates_total;i++)
     {
      //--- запишем время открытия бара в структуру
      TimeToStruct(time[i],dt);
      //--- если день недели этого бара равен invisible_day
      if(dt.day_of_week==invisible_day)
        {
         //--- запишем пустые значения в буферы для этого бара
         ZigZagBuffer1[i]=0;
```

© 2000-2025, MetaQuotes Ltd.

```
         ZigZagBuffer2[i]=0;
        }
      //--- если день недели подходящий, то заполняем буферы
      else
        {
         //--- если номер бара четный
         if(i%2==0)
           {
            //---  пишем в 1-ый буфер High, во 2-ой Low
            ZigZagBuffer1[i]=high[i];
            ZigZagBuffer2[i]=low[i];
           }
         //--- номер бара нечетный
         else
           {
            //--- заполняем бар в обратном порядке
            ZigZagBuffer1[i]=low[i];
            ZigZagBuffer2[i]=high[i];
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид отрезков в зигзаге                          |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах ZigZag
   string comm="";
//--- блок изменения цвета зигзага
   int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
   int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
   comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения толщины линии
   number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5;    // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
```

© 2000-2025, MetaQuotes Ltd.

```
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- добавим информацию о дне, который пропускается в расчетах
   comm="\r\nНеотрисовываемый день - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+c
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_FILLING

Стиль DRAW_FILLING рисует цветную область между значениями двух индикаторных буферов. Фактически этот стиль рисует две линии и закрашивает пространство между ними одним из двух заданных цветов. Предназначен для создания индикаторов, рисующих каналы. Ни один из буферов не может содержать только пустые значения, так как в этом случае отрисовка не происходит.

Можно задавать два цвета заливки:

- первый цвет для тех областей, где значения в первом индикаторном буфере больше значений во втором индикаторном буфере;
- второй цвет для тех областей, где значения во втором индикаторном буфере больше значений в первом индикаторном буфере.

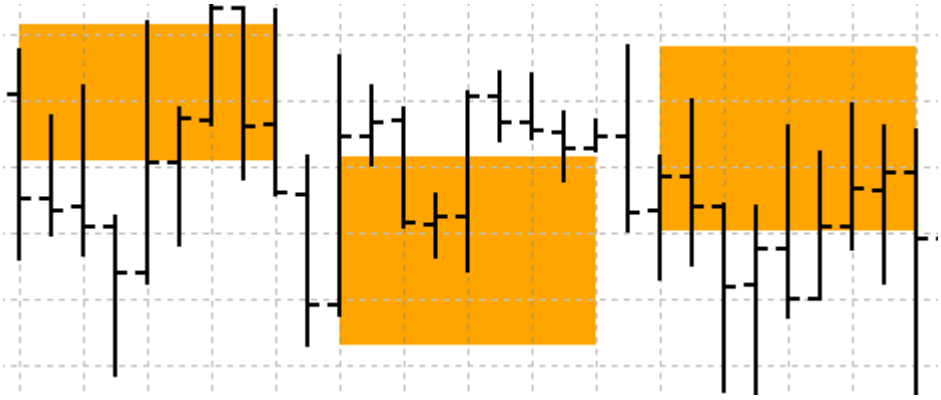
Цвет заливки можно задавать директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет "оживить" индикаторы, чтобы они меняли свой вид в зависимости от текущей ситуации.

Индикатор рассчитывается для всех баров, для которых значения обоих индикаторных буферов не равны 0 и не равны пустому значению. Чтобы указать, какое значение следует считать "пустым", установите это значение в свойстве PLOT_EMPTY_VALUE:

```
   #define INDICATOR_EMPTY_VALUE -1.0
   ...
//--- значение INDICATOR_EMPTY_VALUE (пустое значение) не будет участвовать в расчете
   PlotIndexSetDouble(индекс_построения_DRAW_FILLING,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_
```

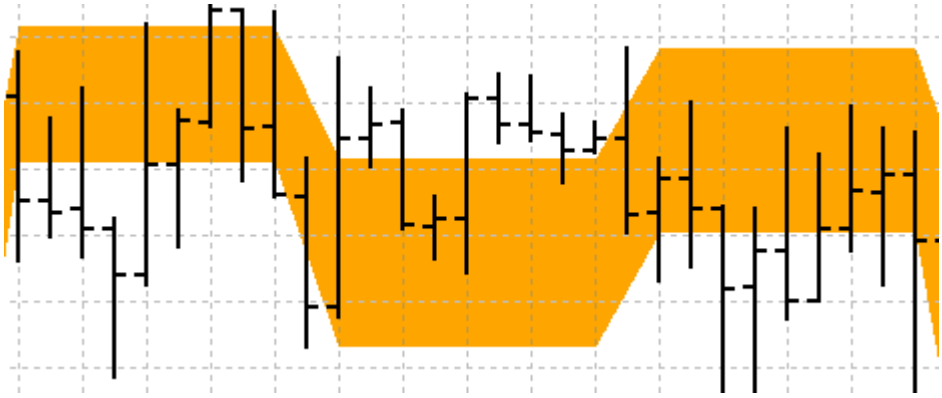
Отрисовка на барах, которые не участвуют в расчете индикатора, будет зависеть от значений в индикаторных буферах:

- Бары, для которых значения обоих индикаторных буферов равны 0, не участвуют в отрисовке индикатора. То есть область с нулевыми значениями не будет закрашиваться.



- Бары, для которых значения индикаторных буферов равны "пустому значению", участвуют в отрисовке индикатора. Область с пустыми значениями будет закрашиваться таким образом, чтобы соединять области со значащими значениями.

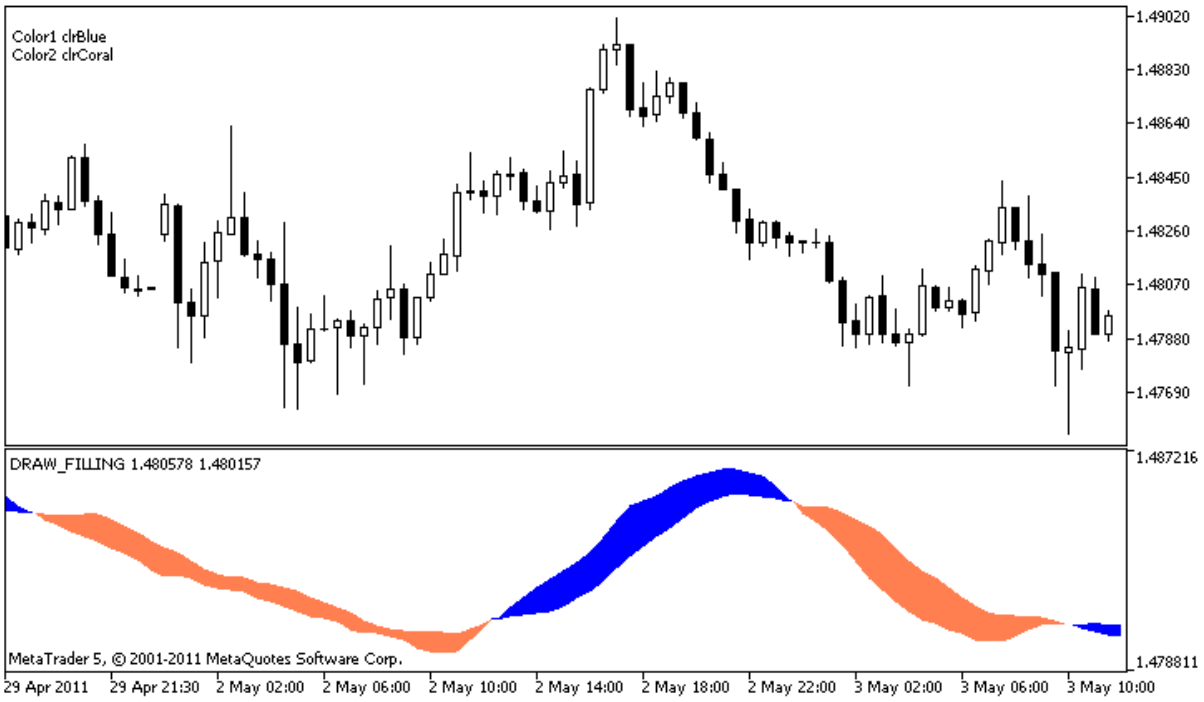
© 2000-2025, MetaQuotes Ltd.



При этом важно отметить, что если "пустое значение" равняется нулю, то бары, не участвующие в расчете индикатора, также будут закрашиваться.

Количество требуемых буферов для построения DRAW_FILLING — 2.

Пример индикатора, рисующего в отдельном окне канал между двумя скользящими средними с разными периодами усреднения. Изменение цвета при пересечении средних визуально показывает смену восходящей и нисходящей тенденций. Цвета меняются случайным образом каждые **N** тиков. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).



Обратите внимание, первоначально для графического построения **plot1** со стилем DRAW_FILLING два цвета задаются с помощью директивы компилятора #property, а затем в функции OnCalculate() новые цвета задаются случайным образом.

```
//+------------------------------------------------------------------+
//|                                                DRAW_FILLING.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_FILLING"
#property description "Рисует в отдельном окне канал между двумя средними"
#property description "Цвет заливки канала меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Intersection
#property indicator_label1  "Intersection"
#property indicator_type1   DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- input параметры
input int      Fast=13;          // период быстрой скользящей средней
input int      Slow=21;          // период медленной скользящей средней
input int      shift=1;          // сдвиг средних в будущее (положительный)
input int      N=5;              // кол-во тиков для изменения
//--- индикаторные буферы
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrCo
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   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);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
```

© 2000-2025, 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[])
  {
   static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось достаточное число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- делаем первый расчет индикатора или данные изменились и требуется полный перерас
   if(prev_calculated==0)
     {
      //--- копируем все значения индикаторов в соответствующие буферы
      int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
     }
   else // экономно заполняем только те данные, которые обновились
     {
      //--- получим разницу в барах между текущим и предыдущим запуском OnCalculate()
      int to_copy=rates_total-prev_calculated;
      //--- если разницы нет, то все равно будем копировать одно значение - на нулевом
      if(to_copy==0) to_copy=1;
      //--- копируем to_copy значений в самый конец индикаторных буферов
      int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет цвета заливки канала                                    |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
```

© 2000-2025, MetaQuotes Ltd.

```
//--- блок изменения цвета линии
   int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
   int size=ArraySize(colors);

//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   int color_index1=number%size;
//--- установим первый цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- запишем первый цвет
   comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   number=MathRand(); // получим случайное число
   int color_index2=number%size;
//--- установим второй цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- запишем второй цвет
   comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_BARS

Стиль DRAW_BARS рисует бары по значениям четырех индикаторных буферов, в которых содержатся цены Open, High, Low и Close. Предназначен для создания собственных индикаторов в виде баров, в том числе в отдельном подокне графика и по другим финансовым инструментам.

Цвет баров можно задавать директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет "оживить" индикаторы, чтобы они меняли свой вид в зависимости от текущей ситуации.

Индикатор рисуется только для тех баров, для которых заданы непустые значения **всех** четырех индикаторных буферов. Чтобы указать, какое значение следует считать "пустым", установите это значение в свойстве PLOT_EMPTY_VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(индекс_построения_DRAW_BARS,PLOT_EMPTY_VALUE,0);
```

Всегда явно заполняйте значениями индикаторные буферы, на пропускаемых барах указывайте в буфере пустое значение.

Количество требуемых буферов для построения DRAW_BARS — 4. Все буферы для построения должны идти последовательно один за другим в заданном порядке: Open, High, Low и Close. Ни один из буферов не может содержать только пустые значения, так как в этом случае отрисовка не происходит.

Пример индикатора, рисующего в отдельном окне бары по указанному финансовому инструменту. Цвет баров меняется случайным образом каждые **N** тиков. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).



© 2000-2025, MetaQuotes Ltd.

Обратите внимание, первоначально для графического построения **plot1** со стилем DRAW_BARS цвет задается с помощью директивы компилятора #property, а затем в функции OnCalculate() цвет выбирается случайным образом из заранее подготовленного списка.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_BARS.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 "Индикатор для демонстрации DRAW_BARS"
#property description "Рисует в отдельном окне бары по выбранному символу"
#property description "Цвет и толщина баров, а также символ, меняются случайным "
#property description "образом через каждые N тиков"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "Bars"
#property indicator_type1   DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input параметры
input int      N=5;              // количество тиков для смены вида
input int      bars=500;         // сколько баров показывать
input bool     messages=false;   // вывод сообщений в лог "Эксперты"
//--- индикаторные буферы
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- имя символа
string symbol;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- если bars слишком мало - досрочно завершаем работу
   if(bars<50)
     {
      Comment("Укажите большее количество баров! Работа индикатора прекращена");
```

© 2000-2025, MetaQuotes Ltd.

```
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//--- имя символа, по которому рисуются бары
   symbol=_Symbol;
//--- установим отображение символа
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- пустое значение
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось достаточное число тиков
   if(ticks>=N)
     {
      //--- выберем новый символ из окна "Обзор рынка"
      symbol=GetRandomSymbolName();
      //--- меняем свойства линии
      ChangeLineAppearance();

      int tries=0;
      //--- сделаем 5 попыток заполнить буферы ценами из symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
         //--- счетчик вызовов функции CopyFromSymbolToBuffers()
         tries++;
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняет индикаторные буферы ценами                             |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total)
  {
//--- в массив rates[] будем копировать цены Open, High, Low и Close
   MqlRates rates[];
//--- счетчик попыток
   int attempts=0;
//--- сколько скопировано
   int copied=0;
//--- делаем 25 попыток получить таймсерию по нужному символу
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- если не удалось скопировать достаточное количество баров
   if(copied!=bars)
     {
      //--- сформируем строку сообщения
      string comm=StringFormat("Для символа %s удалось получить только %d баров из %d
                               name,
                               copied,
                               bars
                               );
      //--- выведем сообщение в комментарий на главное окно графика
      Comment(comm);
      //--- выводим сообщения
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- установим отображение символа
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+name+")");
     }
//--- инициализируем буферы пустыми значениями
   ArrayInitialize(BarsBuffer1,0.0);
   ArrayInitialize(BarsBuffer2,0.0);
   ArrayInitialize(BarsBuffer3,0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayInitialize(BarsBuffer4,0.0);
//--- копируем цены в буферы
   for(int i=0;i<copied;i++)
     {
      //--- вычислим соответствующий индекс для буферов
      int buffer_index=total-copied+i;
      //--- записываем цены в буферы
      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);
  }
//+------------------------------------------------------------------+
//| Возвращает случайным образом символ из Market Watch              |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- количество символов, показываемых в окне "Обзор рынка"
   int symbols=SymbolsTotal(true);
//--- позиция символа в списке - случайное число от 0 до symbols
   int number=MathRand()%symbols;
//--- вернем имя символа по указанной позиции
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид баров                                       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах баров
   string comm="";
//--- блок изменения цвета баров
   int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
   int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
   comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения толщины баров
   number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5;   // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- запишем имя символа
   comm="\r\n"+symbol+comm;

//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_CANDLES

Стиль DRAW_CANDLES рисует японские свечи по значениям четырёх индикаторных буферов, в которых содержатся цены Open, High, Low и Close. Предназначен для создания собственных индикаторов в виде свечей, в том числе в отдельном подокне графика и по другим финансовым инструментам.

Цвет свечей можно задавать директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет "оживить" индикаторы, чтобы они меняли свой вид в зависимости от текущей ситуации.

Индикатор рисуется только для тех баров, для которых заданы непустые значения **всех** четырёх индикаторных буферов. Чтобы указать, какое значение следует считать "пустым", установите это значение в свойстве PLOT_EMPTY_VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(индекс_построения_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

Всегда явно заполняйте значениями индикаторные буферы, на пропускаемых барах указывайте в буфере пустое значение.

Количество требуемых буферов для построения DRAW_CANDLES — 4. Все буферы для построения должны идти последовательно один за другим в заданном порядке: Open, High, Low и Close. Ни один из буферов не может содержать только пустые значения, так как в этом случае отрисовка не происходит.

Для стиля DRAW_CANDLES можно задавать от одного до трёх цветов, в зависимости от этого меняется внешний вид свечей. Если указан только один цвет, то все свечи на графике будут полностью окрашены этим цветом.

```
//--- одинаковые свечи, окрашенные в один цвет
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- указан только один цвет, поэтому все свечи будут одного цвета
#property indicator_color1  clrGreen
```

Если указать два цвета через запятую, то контуры свечи будут отрисовываться первым цветом, а тело вторым цветом.

```
//--- цвет свечей отличается от цвета теней
#property indicator_label1  "Two color candles"
#property indicator_type1   DRAW_CANDLES
//--- тени и контур свечей зеленого цвета, тело белого цвета
#property indicator_color1  clrGreen,clrWhite
```

Для того чтобы по разному показывать растущие и падающие свечи, необходимо указать через запятую 3 цвета. В этом случае контур свечи будет нарисован первым цветом, а цвет бычьей и медвежьей свечей будет задаваться вторым и третьим цветом.

```
//--- цвет свечей отличается от цвета теней
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- тени и контур зелёного цвета, тело бычьей свечи белого цвета, тело медвежьей све
```

© 2000-2025, MetaQuotes Ltd.

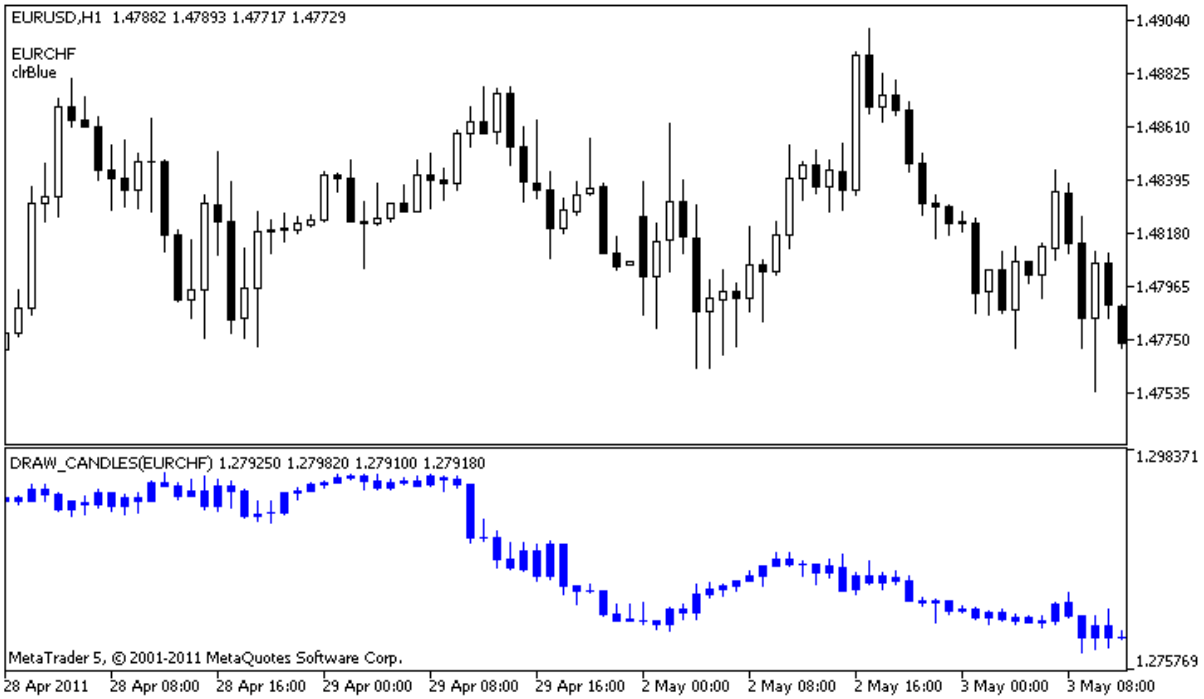
```
#property indicator_color1  clrGreen,clrWhite,clrRed
```

Таким образом, с помощью стиля DRAW_CANDLES можно создавать собственные пользовательские варианты раскраски свечей. Все цвета можно также менять динамически в процессе работы индикатора с помощью функции PlotIndexSetInteger(индекс_построения_DRAW_CANDLES, PLOT_LINE_COLOR, номер_модификатора, цвет) , где номер_модификатора можеть иметь следующие значения:

- 0 - цвет контура и теней
- 1- цвет тела бычьей свечи
- 2 - цвет тела медвежьей свечи

```
//--- установим цвет контура и теней
PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrBlue);
//--- установим цвет тела для бычьей свечи
PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrGreen);
//--- установим цвет тела для медвежьей свечи
PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrRed);
```

Пример индикатора, рисующего в отдельном окне японские свечи по указанному финансовому инструменту. Цвет свечей меняется случайным образом каждые **N** тиков. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).



Обратите внимание, первоначально для графического построения **plot1** цвет задается с помощью директивы компилятора #property, а затем в функции OnCalculate() выбирается новый цвет случайным образом из заранее подготовленного списка.

```
//+------------------------------------------------------------------+
//|                                                DRAW_CANDLES.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_CANDLES."
#property description "Рисует в отдельном окне разным цветом свечи по случайно выбранн
#property description " "
#property description "Цвет и толщина свечей, а также символ, меняются"
#property description "случайным образом через каждые N тиков."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "DRAW_CANDLES1"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- input параметры
input int      N=5;              // количество тиков для смены вида
input int      bars=500;         // сколько баров показывать
input bool     messages=false;   // вывод сообщений в лог "Эксперты"
//--- индикаторные буферы
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- имя символа
string symbol;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- если bars слишком мало - досрочно завершаем работу
   if(bars<50)
     {
      Comment("Укажите большее количество баров! Работа индикатора прекращена");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
//--- пустое значение
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- имя символа, по которому рисуются бары
   symbol=_Symbol;
//--- установим отображение символа
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось достаточное число тиков
   if(ticks>=N)
     {
      //--- выберем новый символ из окна "Обзор рынка"
      symbol=GetRandomSymbolName();
      //--- сменим вид
      ChangeLineAppearance();
      //--- выберем новый символ из окна "Обзор рынка"
      int tries=0;
      //--- сделаем 5 попыток заполнить буферы plot1 ценами из symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
         //--- счетчик вызовов функции CopyFromSymbolToBuffers()
         tries++;
        }
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }
//--- return value of prev_calculated for next call
```

© 2000-2025, MetaQuotes Ltd.

```
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняет указанную свечу                                        |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[]
                             )
  {
//--- в массив rates[] будем копировать цены Open, High, Low и Close
   MqlRates rates[];
//--- счетчик попыток
   int attempts=0;
//--- сколько скопировано
   int copied=0;
//--- делаем 25 попыток получить таймсерию по нужному символу
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- если не удалось скопировать достаточное количество баров
   if(copied!=bars)
     {
      //--- сформируем строку сообщения
      string comm=StringFormat("Для символа %s удалось получить только %d баров из %d
                               name,
                               copied,
                               bars
                               );
      //--- выведем сообщение в комментарий на главное окно графика
      Comment(comm);
      //--- выводим сообщения
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- установим отображение символа
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
     }
//--- инициализируем буферы пустыми значениями
   ArrayInitialize(buff1,0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- на каждом тике копируем цены в буферы
   for(int i=0;i<copied;i++)
     {
      //--- вычислим соответствующий индекс для буферов
      int buffer_index=total-copied+i;
      //--- записываем цены в буферы
      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);
  }
//+------------------------------------------------------------------+
//| Возвращает случайным образом символ из Market Watch              |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- количество символов, показываемых в окне "Обзор рынка"
   int symbols=SymbolsTotal(true);
//--- позиция символа в списке - случайное число от 0 до symbols
   int number=MathRand()%symbols;
//--- вернем имя символа по указанной позиции
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид баров                                       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах баров
   string comm="";
//--- блок изменения цвета баров
   int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
   int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
   int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет
   comm=comm+"\r\n"+(string)colors[color_index];
//--- запишем имя символа
   comm="\r\n"+symbol+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_LINE

Стиль DRAW_COLOR_LINE является цветным вариантом стиля DRAW_LINE, он также рисует линию по значениям индикаторного буфера. Но у этого стиля, как у всех цветных стилей, имеющих в название **COLOR**, есть дополнительный специальный индикаторный буфер, который хранит индекс (номер) цвета из специального заданного массива цветов. Таким образом, цвет каждого участка линии можно задать, если указать индекса цвета, которым линия должна рисоваться на данном баре.

Толщину, стиль и цвета для отображения линии можно задавать как директивами компилятора, так и динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет создавать "живые" индикаторы, которые меняют свой вид в зависимости от текущей ситуации.

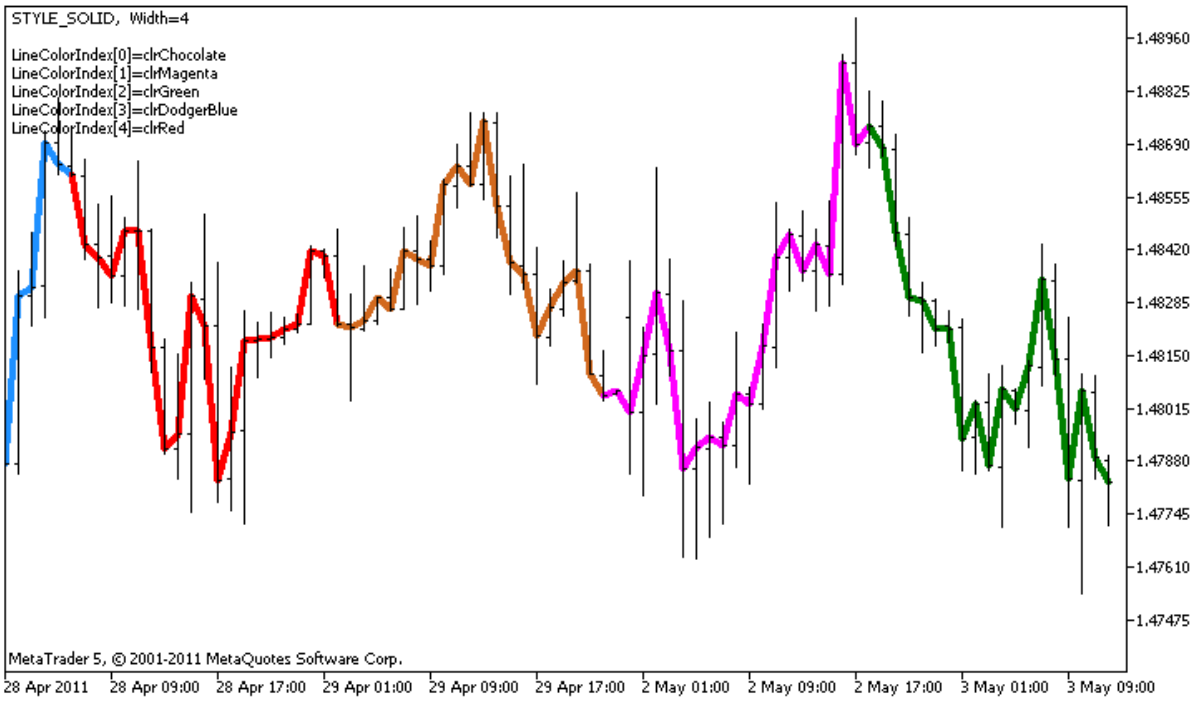
Количество требуемых буферов для построения DRAW_COLOR_LINE — 2:

- один буфер для хранения значений индикатора, по которым рисуется линия;
- один буфер для хранения индекса цвета, которым рисуется линия на каждом баре.

Цвета можно задавать директивой компилятора #property indicator_color1 через запятую. Количество цветов не может превышать 64.

```
//--- зададим 5 цветов для раскраски каждого бара (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (можно ук
```

Пример индикатора, рисующего линию по ценам закрытия баров Close. Толщина и стиль линии меняются случайным образом каждые N=5 тиков.



Цвета, которыми рисуются участки линии, также меняются случайным образом в пользовательской функции ChangeColors().

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Изменяет цвет участков линии                                     |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- количество цветов
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- получим случайное число
      int number=MathRand();
      //--- получим индекс в массиве col[] как остаток от целочисленного деления
      int i=number%size;
      //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  номер графического стиля
                          PLOT_LINE_COLOR,      //  идентификатор свойства
                          plot_color_ind,       //  индекс цвета, куда запишем цвет
                          cols[i]);             //  новый цвет
      //--- запишем цвета
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStrinc
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
```

В примере показана особенность "цветных" версий индикаторов - для смены цвета участка линии не требуется менять значения в буфере ColorLineColors[] (который содержит индексы цветов). Достаточно задать новые цвета в специальном массиве. Это позволяет быстро поменять цвет сразу для всего графического построения, изменяя только небольшой массив цветов с помощью функции PlotIndexSetInteger().

Обратите внимание, первоначально для графического построения **plot1** со стилем DRAW_COLOR_LINE свойства задаются с помощью директивы компилятора #property, а затем в функции OnCalculate() эти три свойства задаются случайным образом.

Параметры N и Length (длина цветных участков в барах) вынесены во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_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"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Индикатор для демонстрации DRAW_COLOR_LINE"
#property description "Рисует цветными кусками по 20 баров линию по ценам Close"
#property description "Толщина,стиль и цвет участков линии меняется случайным"
#property description "образом через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
//--- зададим 5 цветов для раскраски каждого бара (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (можно ук
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input параметры
input int      N=5;           // кол-во тиков для изменения
input int      Length=20;     // длина каждого участка цвета в барах
int            line_colors=5; // количество заданных цветов равно 5 - смотри  выше #pr
//--- буфер для отрисовки
double         ColorLineBuffer[];
//--- буфер для хранения цвета отрисовки линии на каждом баре
double         ColorLineColors[];

//--- массив для хранения цветов содержит 7 элементов
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGolde
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- связывание массива и индикаторного буфера
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- инициализация генератора псевдослучайных чисел
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось критическое число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- меняем цвета, которыми рисуются цветные участки линии
      ChangeColors(colors,5);
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- блок расчета значений индикатора
   for(int i=0;i<rates_total;i++)
     {
      //--- запишем значение индикатора в буфер
      ColorLineBuffer[i]=close[i];
      //--- теперь случайным образом зададим для этого бара  индекс цвета
      int color_index=i%(5*Length);
      color_index=color_index/Length;
      //--- для этого бара линия будет рисоваться цветом, который хранится под номером
      ColorLineColors[i]=color_index;
     }

//--- вернем значение prev_calculated для следующего вызова функции
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет цвет участков линии                                     |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- количество цветов
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- получим случайное число
      int number=MathRand();
      //--- получим индекс в массиве col[] как остаток от целочисленного деления
      int i=number%size;
      //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  номер графического стиля
                          PLOT_LINE_COLOR,      //  идентификатор свойства
                          plot_color_ind,       //  индекс цвета, куда запишем цвет
                          cols[i]);             //  новый цвет
      //--- запишем цвета
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид отображаемой линии в индикаторе             |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
//--- блок изменения толщины линии
   int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5; // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+" Width="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   int size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm=EnumToString(styles[style_index])+", "+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_SECTION

Стиль DRAW_COLOR_SECTION является цветным вариантом стиля DRAW_SECTION, но в отличие от последнего, позволяет рисовать каждую секцию собственным цветом. Стиль DRAW_COLOR_SECTION, как и все цветные стили, имеющие в название **COLOR**, содержит дополнительный специальный индикаторный буфер для хранения индекса (номера) цвета из специального заданного массива цветов. Таким образом, цвет каждой секции можно задать, если указать индекса цвета для бара, на который приходится конец секции.

Толщину, цвет и стиль отрезков можно задавать так же, как и для стиля DRAW_SECTION - директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет "оживить" индикаторы, чтобы они меняли свой вид в зависимости от текущей ситуации.

Секции рисуются от одного непустого значения до другого непустого значения индикаторного буфера, пустые значения пропускаются. Чтобы указать, какое значение следует считать "пустым", установите это значение в свойстве PLOT_EMPTY_VALUE. Например, если индикатор должен рисоваться отрезками по ненулевым значениям, то нужно задать нулевое значение в качестве пустого:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

Всегда явно заполняйте значениями все элементы индикаторного буфера, неотрисовываемым элементам задавайте пустое значение.

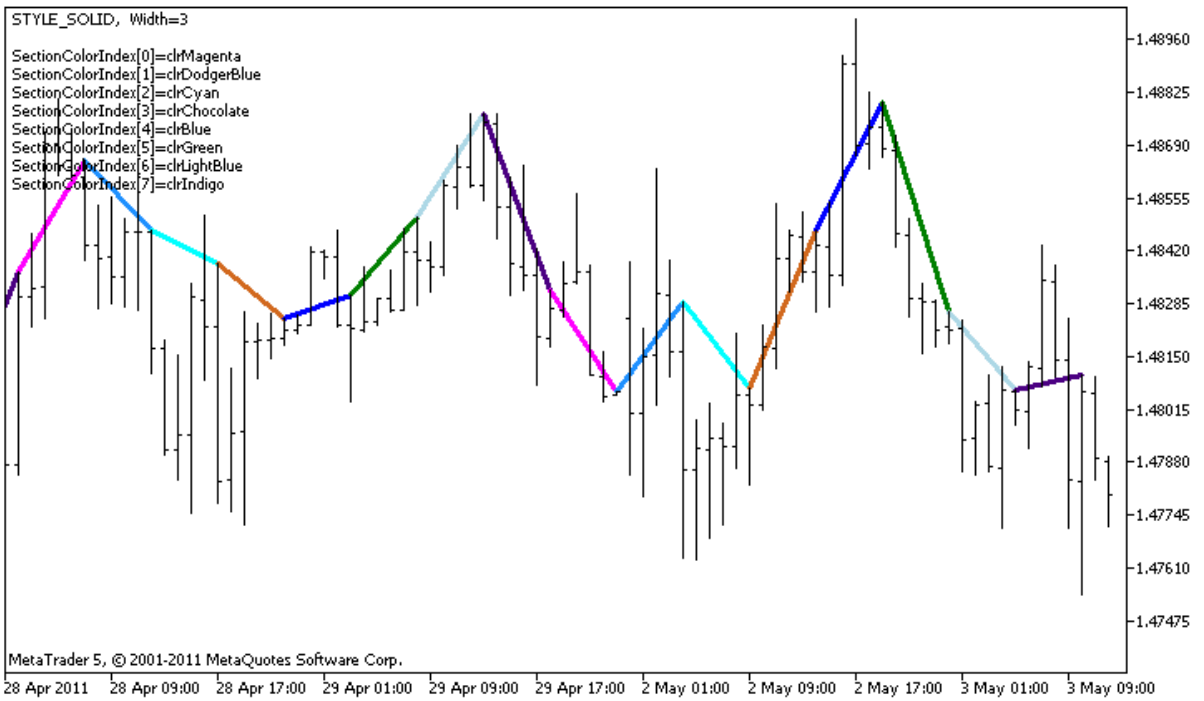
Количество требуемых буферов для построения DRAW_COLOR_SECTION — 2:

- один буфер для хранения значений индикатора, по которым рисуется линия;
- один буфер для хранения индекса цвета, которым рисуется секция (имеет смысл задавать только для непустых значений).

Цвета можно задавать директивой компилятора #property indicator_color1 через запятую. Количество цветов не может превышать 64.

Пример индикатора, рисующего разноцветные секции длиной 5 баров по ценам High. Цвет, толщина и стиль секций меняются случайным образом каждые **N** тиков.

© 2000-2025, MetaQuotes Ltd.



Обратите внимание, первоначально для графического построения **plot1** со стилем DRAW_COLOR_SECTION свойства задается 8 цветов с помощью директивы компилятора #property. Затем в функции OnCalculate() цвета задаются случайным образом из массива цветов colors[].

Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                            DRAW_COLOR_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 "Индикатор для демонстрации DRAW_COLOR_SECTION"
#property description "Рисует цветными отрезками длиной в заданное число баров"
#property description "Цвет, толщина и стиль секций меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorSection
#property indicator_label1  "ColorSection"
#property indicator_type1   DRAW_COLOR_SECTION
//--- зададим 8 цветов для раскраски секций (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,c
#property indicator_style1  STYLE_SOLID
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_width1  1
//--- input параметры
input int      N=5;                      // кол-во тиков для изменения
input int      bars_in_section=5;        // длина секций в барах
//--- вспомогательная переменная для вычисления концов секций
int            divider;
int            color_sections;
//--- буфер для отрисовки
double         ColorSectionBuffer[];
//--- буфер для хранения цвета отрисовки линии на каждом баре
double         ColorSectionColors[];
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- количество цветов для раскраски секций
   color_sections=8;   //  см. комментарий к свойству #property indicator_color1
//--- проверим параметр индикатора
   if(bars_in_section<=0)
     {
      PrintFormat("Недопустимое значение длины секции=%d",bars_in_section);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=color_sections*bars_in_section;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось критическое число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- меняем цвета, которыми рисуются секции
      ChangeColors(colors,color_sections);
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- номер бара, с которого начнем расчет значений индикатора
   int start=0;
//--- если индикатор уже рассчитывали раньше, то установим start на предыдущий бар
   if(prev_calculated>0) start=prev_calculated-1;
//--- здесь все расчеты значений индикатора
   for(int i=start;i<rates_total;i++)
     {
      //--- если номер бара делится без остатка на длину_секции, значит это конец секц
      if(i%bars_in_section==0)
        {
         //--- конец отрезка установим на цену High этого бара
         ColorSectionBuffer[i]=high[i];
         //--- остаток от деления номера бара на длина_секции*количество_цветов
         int rest=i%divider;
         //получим номер цвета =  от 0 до количество_цветов-1
         int color_indext=rest/bars_in_section;
         ColorSectionColors[i]=color_indext;
        }
      //--- если остаток от деления равен bars,
      else
        {
         //--- если ничего не подошло, то этот бар пропускаем - ставим значение 0
         ColorSectionBuffer[i]=0;
        }
     }
//--- вернем значение prev_calculated для следующего вызова функции
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Изменяет цвет участков линии                                     |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- количество цветов
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- получим случайное число
      int number=MathRand();
      //--- получим индекс в массиве col[] как остаток от целочисленного деления
      int i=number%size;
      //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  номер графического стиля
                          PLOT_LINE_COLOR,      //  идентификатор свойства
                          plot_color_ind,       //  индекс цвета, куда запишем цвет
                          cols[i]);             //  новый цвет
      //--- запишем цвета
      comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToSt
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид отображаемой линии в индикаторе             |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
//--- блок изменения толщины линии
   int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5; // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+" Width="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   int size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm=EnumToString(styles[style_index])+", "+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM

Стиль DRAW_COLOR_HISTOGRAM рисует гистограмму цветными столбиками от нуля до указанного значения. Значения берутся из индикаторного буфера. Каждый столбик может иметь свой собственный цвет из заранее предопределенного набора цветов.

Толщину, цвет и стиль гистограммы можно задавать так же, как и для стиля DRAW_HISTOGRAM - директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет изменять вид гистограммы в зависимости от текущей ситуации.

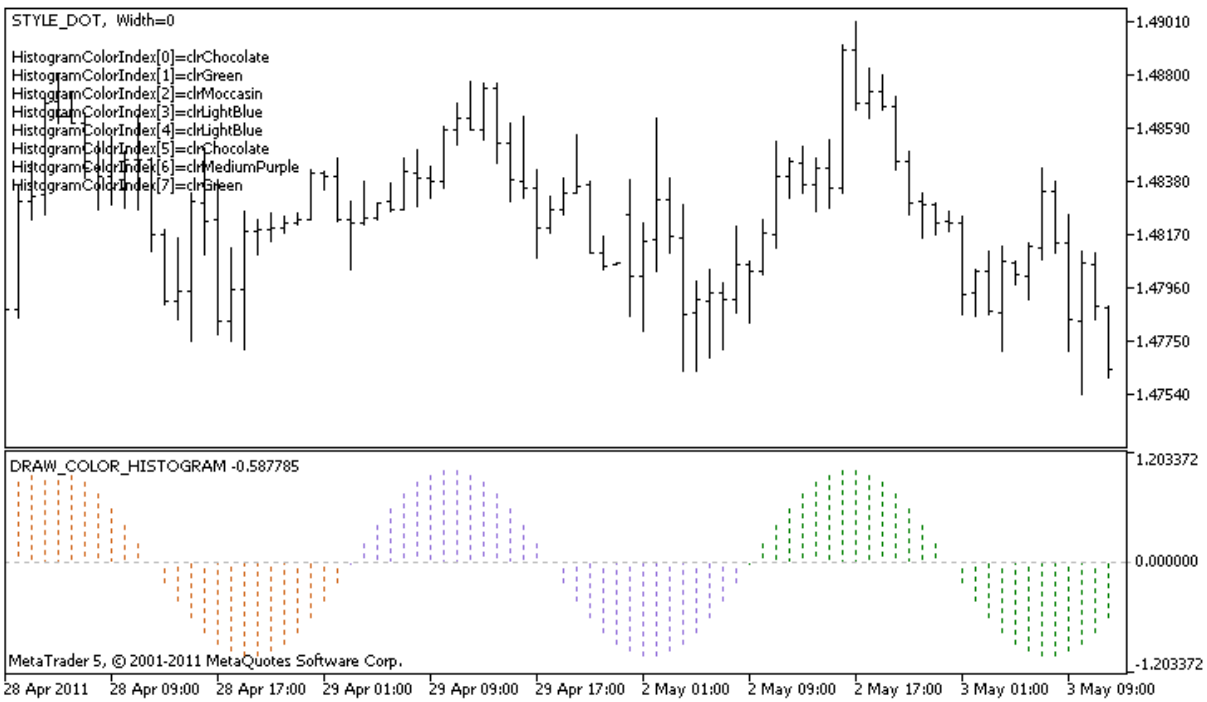
Так как на каждом баре рисуется столбик от нулевого уровня, то DRAW_COLOR_HISTOGRAM лучше использовать для отображения в отдельном подокне графика. Чаще всего этот тип графического построения используется для создания индикаторов осцилляторного типа, например, Awesome Oscillator или Market Facilitation Index. Для пустых неотображаемых значений достаточно указывать нулевое значение.

Количество требуемых буферов для построения DRAW_COLOR_HISTOGRAM — 2:

- один буфер для хранения ненулевого значения вертикального отрезка на каждом баре, второй конец отрезка всегда находится на нулевой линии индикатора;
- один буфер для хранения индекса цвета, которым рисуется секция (имеет смысл задавать только для непустых значений).

Цвета можно задавать директивой компилятора #property indicator_color1 через запятую. Количество цветов не может превышать 64.

Пример индикатора, рисующего заданным цветом синусоиду по функции MathSin(). Цвет, толщина и стиль **всех** столбиков гистограммы меняются случайным образом каждые **N** тиков. Параметр bars задает период синусоиды, то есть через заданное количество баров синусоида будет повторяться по циклу.



© 2000-2025, MetaQuotes Ltd.

Обратите внимание, что первоначально для графического построения **plot1** со стилем DRAW_COLOR_HISTOGRAM задается 5 цветов с помощью директивы компилятора #property indicator_color1, а затем в функции OnCalculate() эти цвета выбираются случайным образом из 14 цветов, хранящихся в массиве colors[]. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_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 "Индикатор для демонстрации DRAW_COLOR_HISTOGRAM"
#property description "Рисует синусоиду гистограммой в отдельном окне"
#property description "Цвет и толщина столбиков меняется случайным образом"
#property description "через каждые N тиков"
#property description "Параметр bars задает количество баров для повторяемости синусои

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- input параметры
input int      bars=30;          // период синусоиды в барах
input int      N=5;              // кол-во тиков для изменения гистограммы
//--- plot Color_Histogram
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- зададим 8 цветов для раскраски секций (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMe
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- буфер значений
double         Color_HistogramBuffer[];
//--- буфер индексов цветов
double         Color_HistogramColors[];
//--- множитель для получения угла 2Pi в радианах при умножении на параметр bars
double         multiplier;
int            color_sections;
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//---- количество цветов для раскраски синусоиды
   color_sections=8;   //  см. комментарий к свойству #property indicator_color1
//--- вычислим множитель
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Задайте значение bars=%d больше 1",bars);
      //--- досрочное прекращение работы индикатора
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось критическое число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- меняем цвета, которыми рисуется гистограмма
      ChangeColors(colors,color_sections);
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- вычисления значений индикатора
   int start=0;
//--- если расчет уже производился на предыдущем запуске OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // установим начало расчетов с предп
//--- заполняем индикаторный буфер значениями
   for(int i=start;i<rates_total;i++)
     {
      //--- значение
      Color_HistogramBuffer[i]=sin(i*multiplier);
      //--- цвет
      int color_index=i%(bars*color_sections);
      color_index/=bars;
      Color_HistogramColors[i]=color_index;
     }
//--- вернем значение prev_calculated для следующего вызова функции
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет цвет участков линии                                     |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- количество цветов
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- получим случайное число
      int number=MathRand();
      //--- получим индекс в массиве col[] как остаток от целочисленного деления
      int i=number%size;
      //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  номер графического стиля
                          PLOT_LINE_COLOR,      //  идентификатор свойства
                          plot_color_ind,       //  индекс цвета, куда запишем цвет
                          cols[i]);             //  новый цвет
      //--- запишем цвета
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид отображаемой линии в индикаторе             |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
//--- блок изменения толщины линии
   int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5; // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+" Width="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   int size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm=EnumToString(styles[style_index])+", "+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM2

Стиль DRAW_COLOR_HISTOGRAM2 рисует заданным гистограмму – вертикальные отрезки по значениям двух индикаторных буферов. Но в отличие от одноцветного DRAW_HISTOGRAM2 в этом стиле каждому столбику гистограммы можно задать свой собственный цвет из предопределенного набора. Значения концов отрезков берутся из индикаторного буфера.

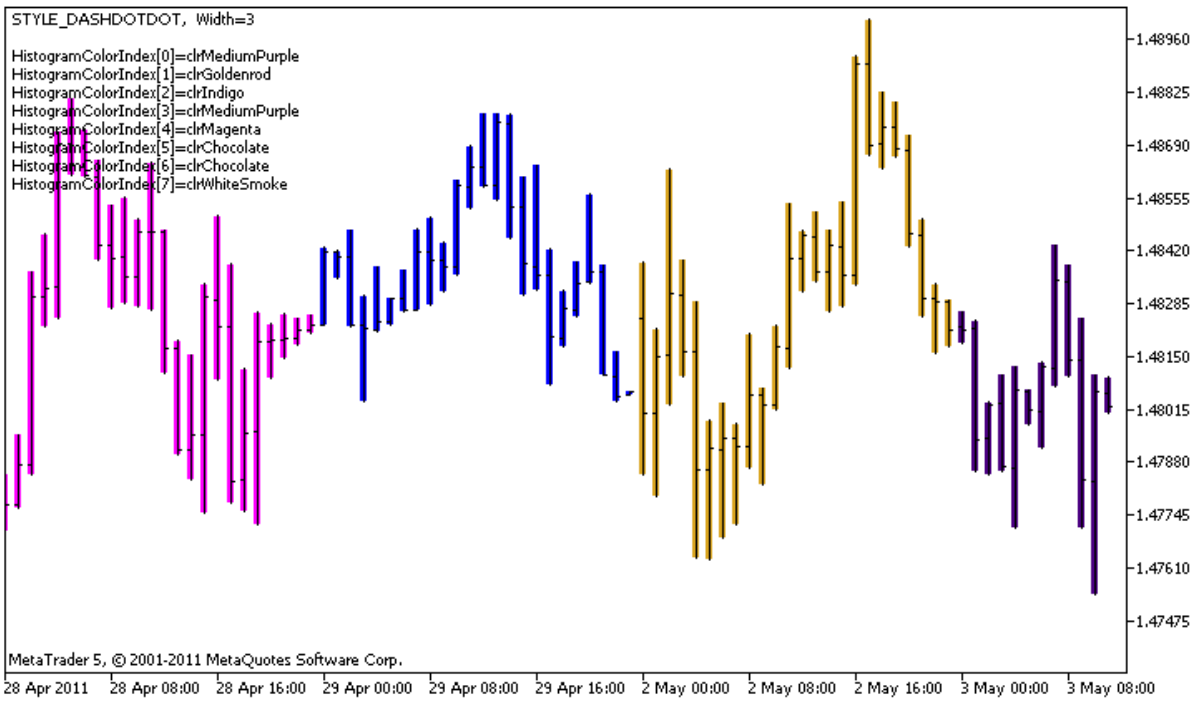
Толщину, стиль и цвета гистограммы можно задавать так же, как и для стиля DRAW_HISTOGRAM2 – директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет изменять вид гистограммы в зависимости от текущей ситуации.

Стиль DRAW_COLOR_HISTOGRAM2 можно использовать как в отдельном подокне графика, так и в главном окне. Для пустых значений отрисовка не производится, все значения в индикаторных буферах нужно устанавливать явным образом. Инициализация буферов пустым значением не производится.

Количество требуемых буферов для построения DRAW_COLOR_HISTOGRAM2 — 3:

- два буфера для хранения верхнего и нижнего конца вертикального отрезка на каждом баре;
- один буфер для хранения индекса цвета, которым рисуется отрезок (имеет смысл задавать только для непустых значений).

Пример индикатора, рисующего заданным цветом гистограмму между ценами High и Low. Для каждого дня недели линии гистограммы рисуются своим цветом. Цвет каждого дня, толщина и стиль гистограммы меняются случайным образом каждые N тиков.



Обратите внимание, что первоначально для графического построения **plot1** со стилем DRAW_COLOR_HISTOGRAM2 задается 5 цветов с помощью директивы компилятора #property indicator_color1, а затем в функции OnCalculate() цвета выбираются случайным образом из 14 цветов, хранящихся в массиве colors[].

© 2000-2025, MetaQuotes Ltd.

Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                        DRAW_COLOR_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 "Индикатор для демонстрации DRAW_COLOR_HISTOGRAM2"
#property description "Рисует на каждом баре отрезок между Open и Close"
#property description "Цвет, толщина и стиль меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot ColorHistogram_2
#property indicator_label1  "ColorHistogram_2"
#property indicator_type1   DRAW_COLOR_HISTOGRAM2
//--- зададим 5 цветов для раскраски гистограммы по дням недели (они хранятся в специа
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- input параметр
input int      N=5;              // кол-во тиков для изменения гистограммы
int            color_sections;
//--- буферы значений
double         ColorHistogram_2Buffer1[];
double         ColorHistogram_2Buffer2[];
//--- буфер индексов цвета
double         ColorHistogram_2Colors[];
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
//--- установим пустое значение
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- количество цветов для раскраски синусоиды
   color_sections=8;   //  см. комментарий к свойству #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось критическое число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- меняем цвета, которыми рисуются гистограмма
      ChangeColors(colors,color_sections);
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- вычисления значений индикатора
   int start=0;
//--- для получения дня недели по времени открытия каждого бара
   MqlDateTime dt;
//--- если расчет уже производился на предыдущем запуске OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // установим начало расчетов с предп
//--- заполняем индикаторный буфер значениями
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- значение
      ColorHistogram_2Buffer1[i]=high[i];
      ColorHistogram_2Buffer2[i]=low[i];
      //--- зададим индекс цвета по дню недели
      int day=dt.day_of_week;
      ColorHistogram_2Colors[i]=day;
     }
//--- вернем значение prev_calculated для следующего вызова функции
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет цвет участков линии                                     |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- количество цветов
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- получим случайное число
      int number=MathRand();
      //--- получим индекс в массиве col[] как остаток от целочисленного деления
      int i=number%size;
      //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  номер графического стиля
                          PLOT_LINE_COLOR,      //  идентификатор свойства
                          plot_color_ind,       //  индекс цвета, куда запишем цвет
                          cols[i]);             //  новый цвет
      //--- запишем цвета
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид отображаемой линии в индикаторе             |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
//--- блок изменения толщины линии
   int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5; // толщина задается от 0 до 4
```

© 2000-2025, MetaQuotes Ltd.

```
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+" Width="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   int size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm=EnumToString(styles[style_index])+", "+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_ARROW

Стиль DRAW_COLOR_ARROW рисует на графике цветом стрелки (символы из набора Wingdings) по значению индикаторного буфера. В отличие от стиля DRAW_ARROW, в нем можно для каждого символа задавать цвет из предопределенного набора цветов, задаваемых свойством indicator_color1.

Толщину и цвет символов можно задавать так же, как и для стиля DRAW_ARROW - директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет изменять вид индикатора в зависимости от текущей ситуации.

Код символа для вывода на график задается с помощью свойства PLOT_ARROW.

```
//--- зададим код символа из шрифта Wingdings для отрисовки в PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

По умолчанию значение PLOT_ARROW=159 (кружок).

Каждая стрелка фактически представляет собой символ, который имеет высоту и точку привязки, и может закрывать собою некоторую важную информацию на графике (например, цену закрытия на баре). Поэтому можно дополнительно указать вертикальное смещение в пикселях, которое не зависит от масштаба графика. На это указанное количество пикселей стрелки будут визуально смещены по вертикали, хотя сами значения индикатора останутся те же самые:

```
//--- зададим смещение стрелок по вертикали в пикселях
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

Отрицательное значение PLOT_ARROW_SHIFT означает смещение стрелок вверх, положительно значения смещает стрелки вниз.

Стиль DRAW_COLOR_ARROW можно использовать как в отдельном подокне графика, так и в главном окне. Пустые значения не отрисовываются и не отображаются в "Окне данных", все значения в индикаторных буферах нужно устанавливать явным образом. Инициализация буферов пустым значением не производится.

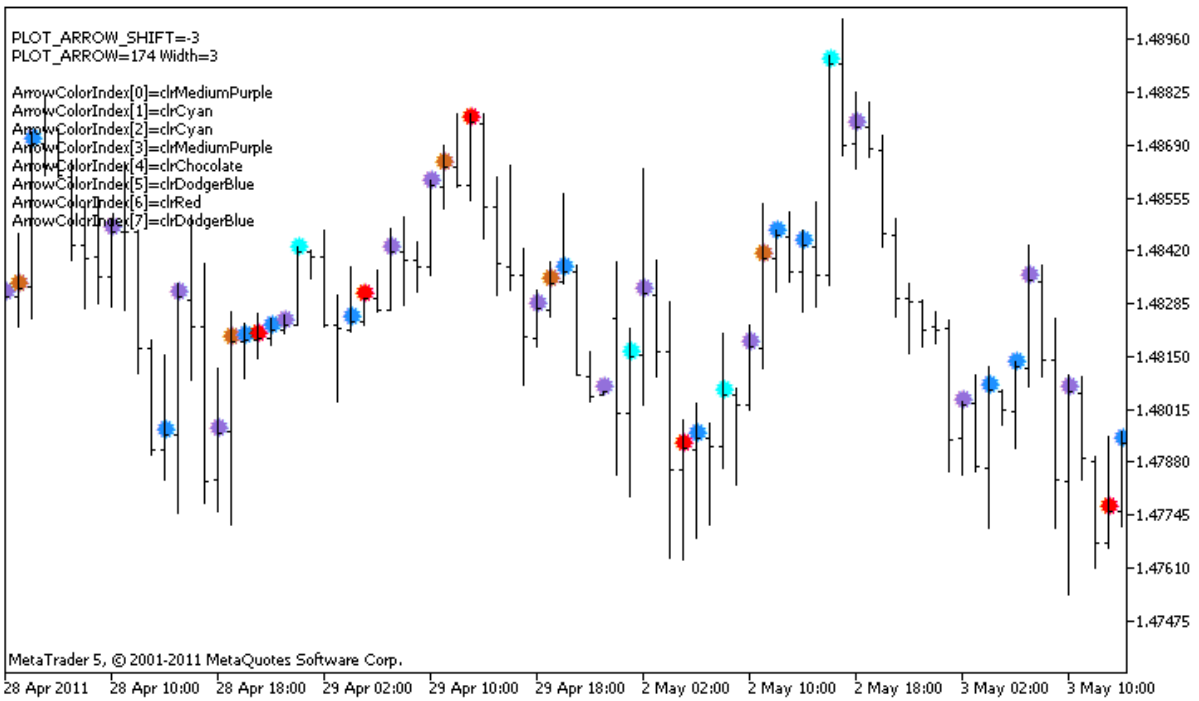
```
//--- установим пустое значение
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_ARROW,PLOT_EMPTY_VALUE,0);
```

Количество требуемых буферов для построения DRAW_COLOR_ARROW — 2:

- один буфер для хранения значения цены, по которой рисуется символ (плюс смещение в пикселях, задаваемое свойством PLOT_ARROW_SHIFT);
- один буфер для хранения индекса цвета, которым рисуется стрелка (имеет смысл задавать только для непустых значений).

Пример индикатора, рисующего стрелки на каждом баре, у которого цена закрытия Close больше цены закрытия предыдущего бара. Толщина, смещение и код символа **всех** стрелок меняются случайным образом каждые **N** тиков. Цвет символа зависит от номера бара, на котором он нарисован.

© 2000-2025, MetaQuotes Ltd.



В примере первоначально для графического построения **plot1** со стилем DRAW_COLOR_ARROW свойства, цвет и размер задаются с помощью директивы компилятора #property, а затем в функции OnCalculate() свойства случайным образом меняются. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

Обратите внимание, первоначально задается 8 цветов с помощью директивы компилятора #property, а затем в функции OnCalculate() цвет выбирается случайным образом из 14 цветов, хранящихся в массиве colors[].

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_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 "Индикатор для демонстрации DRAW_COLOR_ARROW"
#property description "Рисует на графике разным цветом стрелки, задаваемые символами U
#property description "Цвет, размер, смещение и код символа стрелки меняется"
#property description " случайным образом через каждые N тиков"
#property description "Параметр code задает базовое значение: код=159 (кружок)"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorArrow
#property indicator_label1  "ColorArrow"
```

```
#property indicator_type1   DRAW_COLOR_ARROW
//--- зададим 8 цветов для раскраски гистограммы (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagent
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- input параметры
input int       N=5;          // кол-во тиков для изменения
input ushort    code=159;     // код символа для отрисовки в DRAW_ARROW
int             color_sections;
//--- индикаторный буфер для построения
double          ColorArrowBuffer[];
//--- буфер для хранения индексов цвета
double          ColorArrowColors[];
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
//--- зададим код символа для отрисовки в PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- зададим смещение стрелок по вертикали в пикселях
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- установим в качестве пустого значения 0
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- количество цветов для раскраски синусоиды
   color_sections=8;   //  см. комментарий к свойству #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- считаем тики для изменения цвета, размера, смещения и кода стрелки
   ticks++;
//--- если накопилось критическое число тиков
   if(ticks>=N)
     {
      //--- меняем свойства стрелок
      ChangeLineAppearance();
      //--- меняем цвета, которыми рисуются гистограмма
      ChangeColors(colors,color_sections);
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- блок расчета значений индикатора
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- цикл расчета
   for(int i=1;i<rates_total;i++)
     {
      //--- если текущая цена Close больше предыдущей, ставим стрелку
      if(close[i]>close[i-1])
         ColorArrowBuffer[i]=close[i];
      //--- в противном случае указываем нулевое значение
      else
         ColorArrowBuffer[i]=0;
      //--- цвет стрелки
      int index=i%color_sections;
      ColorArrowColors[i]=index;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет цвет участков линии                                     |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- количество цветов
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- получим случайное число
       int number=MathRand();
       //--- получим индекс в массиве col[] как остаток от целочисленного деления
       int i=number%size;
       //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
       PlotIndexSetInteger(0,                    //  номер графического стиля
                           PLOT_LINE_COLOR,      //  идентификатор свойства
                           plot_color_ind,       //  индекс цвета, куда запишем цвет
                           cols[i]);             //  новый цвет
       //--- запишем цвета
       comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStrin
       ChartSetString(0,CHART_COMMENT,comm);
      }
//---
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид отображаемой линии в индикаторе             |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах линии
   string comm="";
//--- блок изменения толщины линии
   int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5; // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+" Width="+IntegerToString(width);

//--- блок изменения кода стрелки (PLOT_ARROW)
   number=MathRand();
//--- получим остаток от целочисленного деления для вычисления нового кода стрелки (от
   int code_add=number%20;
//--- установим новый код символа как сумму code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- запишем код символа PLOT_ARROW
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- блок изменения смещения стрелок по вертикали в пикселях
   number=MathRand();
//--- получим смещение как остаток от целочисленного деления
   int shift=20-number%41;
//--- установим новое смещение
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- запишем смещение PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_ZIGZAG

Стиль DRAW_COLOR_ZIGZAG рисует отрезки разного цвета по значениям двух индикаторных буферов. Этот стиль является цветной версией стиля DRAW_ZIGZAG, то есть позволяет задавать каждому отрезку свой собственный цвет из заранее предопределенного набора цветов. Отрезки рисуются от значения в первом буфере до значения во втором индикаторном буфере. Ни один из буферов не может содержать только пустые значения, так как в этом случае отрисовка не происходит.

Толщину, цвет и стиль отображения линии можно задавать так же, как и для стиля DRAW_ZIGZAG - директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет "оживить" индикаторы, чтобы они меняли свой вид в зависимости от текущей ситуации.

Секции рисуются от одного непустого значения одного буфера до непустого значения другого индикаторного буфера. Чтобы указать, какое значение следует считать "пустым", установите это значение в свойстве PLOT_EMPTY_VALUE:

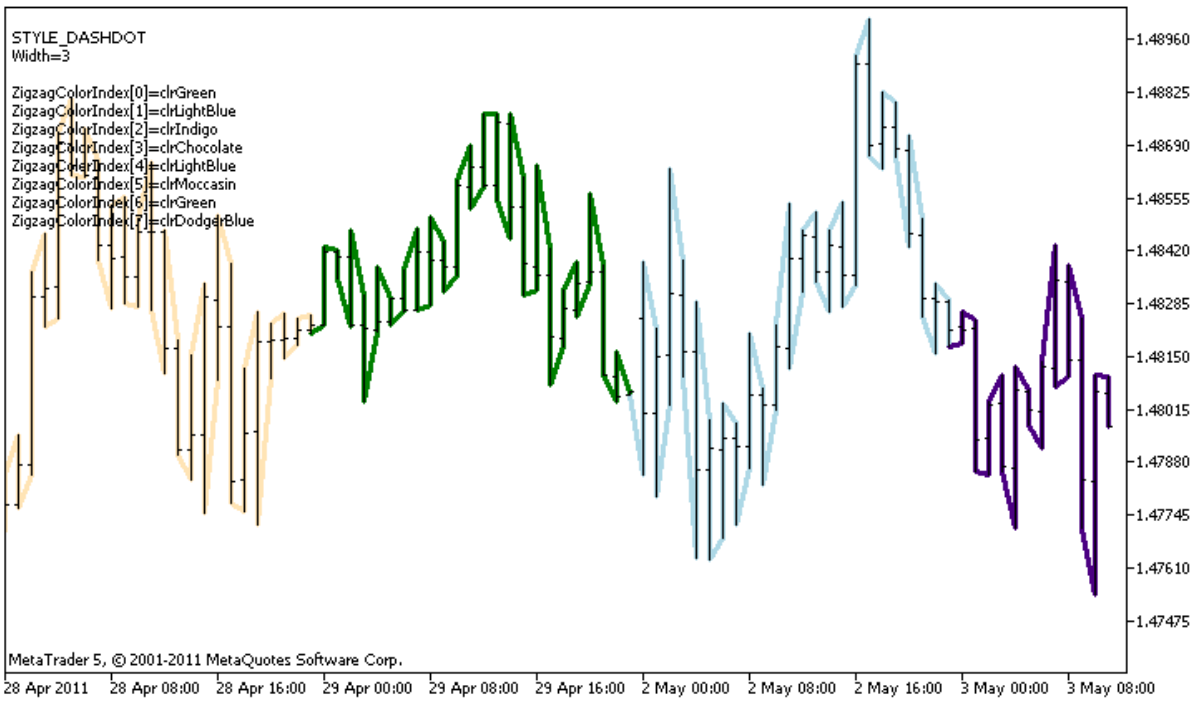
```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Всегда явно заполняйте значениями индикаторные буфера, на пропускаемых барах указывайте в буфере пустое значение.

Количество требуемых буферов для построения DRAW_COLOR_ZIGZAG — 3:

- два буфера для хранения значений концов секций зигзага;
- один буфер для хранения индекса цвета, которым рисуется секция (имеет смысл задавать только для непустых значений).

Пример индикатора, рисующего пилу по ценам High и Low. Цвет, толщина и стиль линий зигзага меняются случайным образом каждые N тиков.



© 2000-2025, MetaQuotes Ltd.

Обратите внимание, первоначально для графического построения **plot1** со стилем DRAW_COLOR_ZIGZAG задается 8 цветов с помощью директивы компилятора #property, а затем в функции OnCalculate() цвет выбирается случайным образом из 14 цветов, хранящихся в массиве colors[].

Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_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 "Индикатор для демонстрации DRAW_COLOR_ZIGZAG"
#property description "Рисует ломанную линию цветными отрезками, цвет зависит от номер
#property description "Цвет, толщина и стиль отрезков меняется случайным"
#property description " образом через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot Color_Zigzag
#property indicator_label1  "Color_Zigzag"
#property indicator_type1   DRAW_COLOR_ZIGZAG
//--- зададим 8 цветов для раскраски секций (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input параметр
input int      N=5;              // кол-во тиков для изменения
int            color_sections;
//--- буферы значений концов отрезков
double         Color_ZigzagBuffer1[];
double         Color_ZigzagBuffer2[];
//--- буфер индексов цвета для концов отрезков
double         Color_ZigzagColors[];
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//---- количество цветов для раскраски зигзага
   color_sections=8;   //  см. комментарий к свойству #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля, цвета и толщины линии
   ticks++;
//--- если накопилось достаточное число тиков
   if(ticks>=N)
     {
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- меняем цвета, которыми рисуются секции
      ChangeColors(colors,color_sections);
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }

//--- структура времени понадобится для получения дня недели каждого бара
   MqlDateTime dt;

//--- позиция начала расчетов
   int start=0;
//--- если индикатор рассчитывался на предыдущем тике, то начинам расчет с предпоследн
   if(prev_calculated!=0) start=prev_calculated-1;
//--- цикл расчетов
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=start;i<rates_total;i++)
     {
      //--- запишем время открытия бара в структуру
      TimeToStruct(time[i],dt);

      //--- если номер бара четный
      if(i%2==0)
        {
         //---  пишем в 1-ый буфер High, во 2-ой Low
         Color_ZigzagBuffer1[i]=high[i];
         Color_ZigzagBuffer2[i]=low[i];
         //--- цвет отрезка
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
      //--- номер бара нечетный
      else
        {
         //--- заполняем бар в обратном порядке
         Color_ZigzagBuffer1[i]=low[i];
         Color_ZigzagBuffer2[i]=high[i];
         //--- цвет отрезка
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Изменяет цвет отрезков зигзага                                   |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- количество цветов
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- получим случайное число
      int number=MathRand();
      //--- получим индекс в массиве col[] как остаток от целочисленного деления
      int i=number%size;
      //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  номер графического стиля
                          PLOT_LINE_COLOR,      //  идентификатор свойства
                          plot_color_ind,       //  индекс цвета, куда запишем цвет
                          cols[i]);             //  новый цвет
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- запишем цвета
      comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид отрезков в зигзаге                          |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах Color_ZigZag
   string comm="";
//--- блок изменения толщины линии
   int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5;   // толщина задается от о до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
   number=MathRand();
//--- делитель числа равен размеру массива styles
   int size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
   int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_BARS

Стиль DRAW_COLOR_BARS рисует бары по значениям четырех индикаторных буферов, в которых содержатся цены Open, High, Low и Close. Этот стиль является продвинутой версией стиля DRAW_BARS и позволяет задавать для каждого бара свой цвет из заранее предопределенного набора цветов. Предназначен для создания собственных индикаторов в виде баров, в том числе в отдельном подокне графика и по другим финансовым инструментам.

Цвет баров можно задавать директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет "оживить" индикаторы, чтобы они меняли свой вид в зависимости от текущей ситуации.

Индикатор рисуется только для тех баров, для которых заданы непустые значения **всех** четырех индикаторных буферов. Чтобы указать, какое значение следует считать "пустым", установите это значение в свойстве PLOT_EMPTY_VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

Всегда явно заполняйте значениями индикаторные буферы, на пропускаемых барах указывайте в буфере пустое значение.

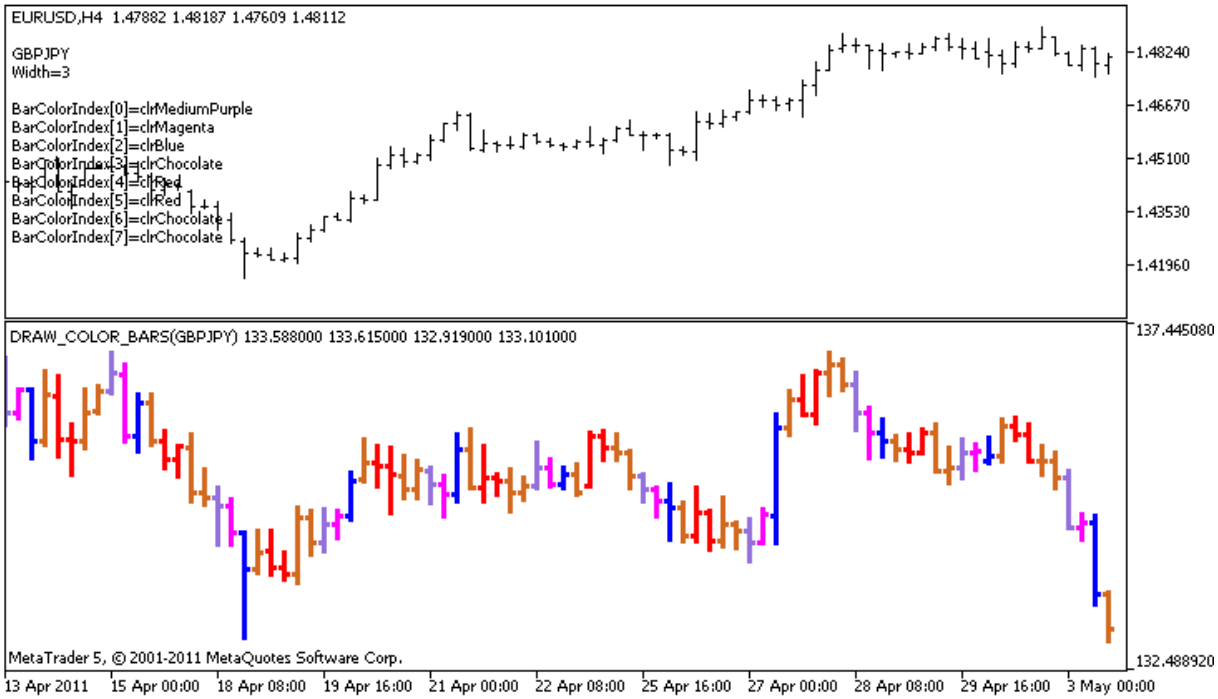
Количество требуемых буферов для построения DRAW_COLOR_BARS — 5:

- четыре буфера для хранения значений Open, High, Low и Close;
- один буфер для хранения индекса цвета, которым рисуется бар (имеет смысл задавать только отрисовываемых баров).

Все буферы для построения должны идти последовательно один за другим в заданном порядке: Open, High, Low, Close и буфер цвета. Ни один из ценовых буферов не может содержать только пустые значения, так как в этом случае отрисовка не происходит.

Пример индикатора, рисующего в отдельном окне бары по указанному финансовому инструменту. Цвет баров меняется случайным образом каждые **N** тиков. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

© 2000-2025, MetaQuotes Ltd.



Обратите внимание, первоначально для графического построения **plot1** со стилем DRAW_COLOR_BARS задается 8 цветов с помощью директивы компилятора #property, а затем в функции OnCalculate() цвет выбирается случайным образом из 14 цветов, хранящихся в массиве colors[].

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_BARS.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 "Индикатор для демонстрации DRAW_COLOR_BARS"
#property description "Рисует в отдельном окне разным цветом бары по выбранному симво
#property description "Цвет и толщина баров, а также символ, меняются случайным "
#property description "образом через каждые N тиков"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorBars
#property indicator_label1  "ColorBars"
#property indicator_type1   DRAW_COLOR_BARS
//--- зададим 8 цветов для раскраски баров (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input параметры
```

```
input int      N=5;              // количество тиков для смены вида
input int      bars=500;         // сколько баров показывать
input bool     messages=false;   // вывод сообщений в лог "Эксперты"
//--- индикаторные буферы
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- имя символа
string symbol;
int    bars_colors;
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   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);
//---- количество цветов для раскраски баров
   bars_colors=8;   //  см. комментарий к свойству #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- считаем тики для изменения стиля, цвета и толщины бара
   ticks++;
//--- если накопилось достаточное число тиков
   if(ticks>=N)
     {
      //--- выберем новый символ из окна "Обзор рынка"
      symbol=GetRandomSymbolName();
      //--- меняем свойства линии
      ChangeLineAppearance();
      //--- меняем цвета, которыми рисуются бары
      ChangeColors(colors,bars_colors);
      int tries=0;
      //--- сделаем 5 попыток заполнить буферы ценами из symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
         //--- счетчик вызовов функции CopyFromSymbolToBuffers()
         tries++;
        }
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы ценами                             |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
  {
//--- в массив rates[] будем копировать цены Open, High, Low и Close
   MqlRates rates[];
//--- счетчик попыток
   int attempts=0;
//--- сколько скопировано
   int copied=0;
//--- делаем 25 попыток получить таймсерию по нужному символу
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- если не удалось скопировать достаточное количество баров
   if(copied!=bars)
     {
      //--- сформируем строку сообщения
      string comm=StringFormat("Для символа %s удалось получить только %d баров из %d
                               name,
                               copied,
```

© 2000-2025, MetaQuotes Ltd.

```
                                bars
                                );
      //--- выведем сообщение в комментарий на главное окно графика
      Comment(comm);
      //--- выводим сообщения
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- установим отображение символа
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS("+name+")");
     }
//--- инициализируем буферы пустыми значениями
   ArrayInitialize(ColorBarsBuffer1,0.0);
   ArrayInitialize(ColorBarsBuffer2,0.0);
   ArrayInitialize(ColorBarsBuffer3,0.0);
   ArrayInitialize(ColorBarsBuffer4,0.0);

//--- копируем цены в буферы
   for(int i=0;i<copied;i++)
     {
      //--- вычислим соответствующий индекс для буферов
      int buffer_index=total-copied+i;
      //--- записываем цены в буферы
      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);
  }
//+------------------------------------------------------------------+
//| Возвращает случайным образом символ из Market Watch              |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- количество символов, показываемых в окне "Обзор рынка"
   int symbols=SymbolsTotal(true);
//--- позиция символа в списке - случайное число от 0 до symbols
   int number=MathRand()%symbols;
//--- вернем имя символа по указанной позиции
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Изменяет цвет отрезков зигзага                                   |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- количество цветов
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- получим случайное число
      int number=MathRand();
      //--- получим индекс в массиве col[] как остаток от целочисленного деления
      int i=number%size;
      //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  номер графического стиля
                          PLOT_LINE_COLOR,      //  идентификатор свойства
                          plot_color_ind,       //  индекс цвета, куда запишем цвет
                          cols[i]);             //  новый цвет
      //--- запишем цвета
      comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид баров                                       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах баров
   string comm="";

//--- блок изменения толщины баров
   int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
   int width=number%5;    // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- запишем имя символа
   comm="\r\n"+symbol+comm;

//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_CANDLES

Стиль DRAW_COLOR_CANDLES как и DRAW_CANDLES рисует японские свечи по значениям четырех индикаторных буферов, в которых содержатся цены Open, High, Low и Close. Но кроме этого он позволяет задавать цвет для каждой свечи из заданного набора. Для этого в стиль добавлен специальный цветовой буфер, который хранит индексы цветов для каждого бара. Предназначен для создания собственных индикаторов в виде свечей, в том числе в отдельном подокне графика и по другим финансовым инструментам.

Количество цветов для раскраски свечей можно задавать директивами компилятора или динамически с помощью функции PlotIndexSetInteger(). Динамическое изменение свойств графического построения позволяет "оживить" индикаторы, чтобы они меняли свой вид в зависимости от текущей ситуации.

Индикатор рисуется только для тех баров, для которых заданы непустые значения четырех индикаторных буферов для хранения цен. Чтобы указать, какое значение следует считать "пустым", установите это значение в свойстве PLOT_EMPTY_VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_CANDLES,PLOT_EMPTY_VALUE,0);
```

Всегда явно заполняйте значениями индикаторные буферы, на пропускаемых барах указывайте в буфере пустое значение.

Количество требуемых буферов для построения DRAW_COLOR_CANDLES — 5:

- четыре буфера для хранения значений Open, High, Low и Close;
- один буфер для хранения индекса цвета, которым рисуется свеча (имеет смысл задавать только отрисовываемых свечей).

Все буферы для построения должны идти последовательно один за другим в заданном порядке: Open, High, Low, Close и буфер цвета. Ни один из ценовых буферов не может содержать только пустые значения, так как в этом случае отрисовка не происходит.

Пример индикатора, рисующего в отдельном окне японские свечи по указанному финансовому инструменту. Цвет свечей меняется случайным образом каждые **N** тиков. Параметр N вынесен во внешние параметры индикатора для возможности ручной установки (закладка "Параметры" в окне свойств индикатора).

© 2000-2025, MetaQuotes Ltd.



Обратите внимание, первоначально для графического построения **plot1** цвет задается с помощью директивы компилятора #property, а затем в функции OnCalculate() выбирается новый цвет случайным образом из заранее подготовленного списка.

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_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 "Индикатор для демонстрации DRAW_COLOR_CANDLES."
#property description "Рисует в отдельном окне разным цветом свечи по случайно выбранн
#property description " "
#property description "Цвет и толщина свечей, а также символ, меняются"
#property description "случайным образом через каждые N тиков."

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
//--- зададим 8 цветов для раскраски свечей (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//--- input параметры
input int      N=5;              // количество тиков для смены вида
input int      bars=500;         // сколько свечей показывать
input bool     messages=false;   // вывод сообщений в лог "Эксперты"
//--- индикаторные буферы
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- имя символа
string symbol;
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- если bars слишком мало - досрочно завершаем работу
   if(bars<50)
     {
      Comment("Укажите большее количество баров! Работа индикатора прекращена");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   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);
//--- пустое значение
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- имя символа, по которому рисуются бары
   symbol=_Symbol;
//--- установим отображение символа
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
//---- количество цветов для раскраски свечей
   candles_colors=8;     //  см. комментарий к свойству #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики для изменения стиля и цвета
   ticks++;
//--- если накопилось достаточное число тиков
   if(ticks>=N)
     {
      //--- выберем новый символ из окна "Обзор рынка"
      symbol=GetRandomSymbolName();
      //--- сменим вид
      ChangeLineAppearance();
      //--- меняем цвета, которыми рисуются бары
      ChangeColors(colors,candles_colors);

      int tries=0;
      //--- сделаем 5 попыток заполнить буферы plot1 ценами из symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
         //--- счетчик вызовов функции CopyFromSymbolToBuffers()
         tries++;
        }
      //--- сбрасываем счетчик тиков в ноль
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняет указанную свечу                                        |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
```

© 2000-2025, MetaQuotes Ltd.

```
                              double &buff2[],
                              double &buff3[],
                              double &buff4[],
                              double &col_buffer[],
                              int    cndl_colors
                              )
  {
//--- в массив rates[] будем копировать цены Open, High, Low и Close
   MqlRates rates[];
//--- счетчик попыток
   int attempts=0;
//--- сколько скопировано
   int copied=0;
//--- делаем 25 попыток получить таймсерию по нужному символу
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- если не удалось скопировать достаточное количество баров
   if(copied!=bars)
     {
      //--- сформируем строку сообщения
      string comm=StringFormat("Для символа %s удалось получить только %d баров из %d
                               name,
                               copied,
                               bars
                               );
      //--- выведем сообщение в комментарий на главное окно графика
      Comment(comm);
      //--- выводим сообщения
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- установим отображение символа
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
     }
//--- инициализируем буферы пустыми значениями
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- на каждом тике копируем цены в буферы
   for(int i=0;i<copied;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- вычислим соответствующий индекс для буферов
      int buffer_index=total-copied+i;
      //--- записываем цены в буферы
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
      //--- зададим цвет свечи
      int color_index=i%cndl_colors;
      col_buffer[buffer_index]=color_index;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Возвращает случайным образом символ из Market Watch              |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- количество символов, показываемых в окне "Обзор рынка"
   int symbols=SymbolsTotal(true);
//--- позиция символа в списке - случайное число от 0 до symbols
   int number=MathRand()%symbols;
//--- вернем имя символа по указанной позиции
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Изменяет цвет отрезков свечей                                    |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- количество цветов
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- получим случайное число
      int number=MathRand();
      //--- получим индекс в массиве col[] как остаток от целочисленного деления
      int i=number%size;
      //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  номер графического стиля
                          PLOT_LINE_COLOR,      //  идентификатор свойства
                          plot_color_ind,       //  индекс цвета, куда запишем цвет
                          cols[i]);             //  новый цвет
      //--- запишем цвета
      comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
```

© 2000-2025, MetaQuotes Ltd.

```
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Изменяет внешний вид свечей                                      |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- строка для формирования информации о свойствах свечей
   string comm="";
//--- запишем имя символа
   comm="\r\n"+symbol+comm;
//--- выведем информацию на график через комментарий
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

Связь между свойствами индикатора и соответствующими функциями

Каждый пользовательский индикатор обладает множеством свойств, часть из которых является обязательной и всегда располагается в самом начале описания. Это свойства:

- указание окна для отображения индикатора - indicator_separate_window или indicator_chart_window;
- количество индикаторных буферов - indicator_buffers;
- количество графических построений,которые выводит индикатора - indicator_plots.

Но есть и другая часть свойств, которые можно задавать как через директивы препроцессора, так и через функции для создания пользовательского индикатора. В таблице перечислены эти свойства и соответствующие им функции.

Директивы для свойства подокна индикатора	Функции типа IndicatorSet...()	Описание устанавливаемого свойства подокна
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT, nHeight)	Значение фиксированной высоты подокна
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM, dMaxValue)	Минимальное значение вертикальной оси
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM, dMinValue)	Максимальное значение вертикальной оси
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE, N-1, nLevelValue)	Значение уровня номер N на вертикальной оси
нет директивы препроцессора	IndicatorSetString (INDICATOR_LEVELTEXT, N-1, sLevelName)	Имя отображаемого уровня
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR, N-1, nLevelColor)	Цвет для отображения уровня номер N
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH, N-1, nLevelWidth)	Толщина линии для отображения уровня номер N
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE, N-1, nLevelStyle)	Стиль линии для отображения уровня номер N
Директивы для свойства графических построений	**Функции типа PlotIndexSet...()**	**Описание устанавливаемого свойства для графического построения**

© 2000-2025, MetaQuotes Ltd.

Директивы для свойства подокна индикатора	Функции типа IndicatorSet...()	Описание устанавливаемого свойства подокна
indicator_labelN	PlotIndexSetString(N-1,PLOT_LABEL,sLabel)	Краткое наименование для графического построения номер N. Показывается в окне DataWindow и во всплывающей подсказке при наведении курсора на построение
indicator_colorN	PlotIndexSetInteger(N-1, PLOT_LINE_COLOR, nColor)	Цвет линии для графического построения номер N
indicator_styleN	PlotIndexSetInteger(N-1, PLOT_LINE_STYLE, nType)	Стиль линии для графического построения номер N
indicator_typeN	PlotIndexSetInteger(N-1, PLOT_DRAW_TYPE, nType)	Тип линии для графического построения номер N
indicator_widthN	PlotIndexSetInteger(N-1, PLOT_LINE_WIDTH, nWidth)	Толщина линии для графического построения номер N
Общие свойства индикатора	**Функции типа IndicatorSet...()**	**Описание**
нет директивы препроцессора	IndicatorSetString (INDICATOR_SHORTNAME, sShortName)	Устанавливает удобное короткое имя индикатора, которое будет отображаться в списке индикаторов (вызывается в терминале сочетанием **Ctrl+I**).
нет директивы препроцессора	IndicatorSetInteger (INDICATOR_DIGITS, nDigits)	Устанавливает требуемую точность для отображения значений индикатора - количество знаков после десятичной точки
нет директивы препроцессора	IndicatorSetInteger (INDICATOR_LEVELS, nLevels)	Задает количество уровней на окне индикатора
indicator_applied_price	Функции нет, свойство устанавливается только директивой препроцессора.	Тип цены по умолчанию, используемый для расчета значений индикатора. Указывается при необходимости только при использовании функции OnCalculate() первого вида. Значение свойства можно также задавать из диалога свойств индикатора на

© 2000-2025, MetaQuotes Ltd.

Директивы для свойства подокна индикатора	Функции типа IndicatorSet...()	Описание устанавливаемого свойства подокна
		вкладке "Параметры" - "Применить к".

Необходимо отметить, что нумерация уровней и графических построений в терминах препроцессора начинается с единицы, в то время как нумерация тех же самых свойств при использовании функций начинается с нуля, то есть нужно указывать значение на 1 меньше, чем мы указали бы с использованием #property.

Есть несколько директив, для которых не существуют соответствующие функции:

Директива	Описание
indicator_chart_window	Индикатор показывается в главном окне
indicator_separate_window	Индикатор показывается в отдельном подокне
indicator_buffers	Указывает количество требуемых индикаторных буферов
indicator_plots	Указывает количество графических построений в индикаторе

Смотри также

Свойства пользовательских индикаторов

© 2000-2025, MetaQuotes Ltd.

SetIndexBuffer

Связывает указанный индикаторный буфер с одномерным динамическим массивом типа double.

```
bool  SetIndexBuffer(
   int                   index,          // индекс буфера
   double                buffer[],       // массив
   ENUM_INDEXBUFFER_TYPE  data_type      // что будем хранить
   );
```

Параметры

index

[in] Номер индикаторного буфера. Нумерация начинается с 0. Номер должен быть меньше значения, объявленного в #property indicator_buffers.

buffer[]

[in] Массив, объявленный в программе пользовательского индикатора.

data_type

[in] Тип данных, хранящихся в индикаторном массиве. По умолчанию INDICATOR_DATA (значения рассчитанного индикатора). Может также принимать значение INDICATOR_COLOR_INDEX, тогда данный буфер предназначен для хранения индексов цветов для предыдущего индикаторного буфера. Можно задать до 64 цветов в строке #property indicator_colorN. Значение INDICATOR_CALCULATIONS означает, что данный буфер участвует в промежуточных расчетах индикатора и не предназначен для отрисовки.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

Примечание

После связывания динамический массив *buffer[]* будет иметь индексацию как в обычных массивах, даже если для связываемого массива будет предварительно установлена индексация как в таймсериях. Если необходимо изменить порядок доступа к элементам индикаторного массива, необходимо применить функцию ArraySetAsSeries() после связывания массива функцией SetIndexBuffer(). При этом необходимо иметь ввиду, что нельзя изменять размер для динамических массивов, назначенных в качестве индикаторных буферов функцией SetIndexBuffer(). Для индикаторных буферов все операции по изменению размера производит исполняющая подсистема терминала.

Пример:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//--- input parameters
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;
//--- indicator buffers
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   if(AsSeries) ArraySetAsSeries(MABuffer,true);
   Print("Индикаторный буфер является таймсерией = ",ArrayGetAsSeries(MABuffer));
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Индикаторный буфер после SetIndexBuffer() является таймсерией = ",
         ArrayGetAsSeries(MABuffer));

//--- изменим порядок доступа к элементам индикаторного буфера
   ArraySetAsSeries(MABuffer,AsSeries);

   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- скопируем значения скользящей средней в буфер MABuffer
   int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

   Print("MABuffer[0] = ",MABuffer[0]);// в зависимости от значения AsSeries
                                       // будем получать либо самое старое значение
```

© 2000-2025, MetaQuotes Ltd.

```
                                        // либо на текущем незавершенном баре
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Смотри также

Свойства пользовательских индикаторов, Доступ к таймсериям и индикаторам

© 2000-2025, MetaQuotes Ltd.

IndicatorSetDouble

Задаёт значение соответствующего свойства индикатора. Свойство индикатора должно быть типа double. Существует 2 варианта функции.

Вызов с указанием идентификатора свойства.

```
bool  IndicatorSetDouble(
   int     prop_id,           // идентификатор
   double  prop_value         // устанавливаемое значение
   );
```

Вызов с указанием идентификатора и модификатора свойства.

```
bool  IndicatorSetDouble(
   int     prop_id,           // идентификатор
   int     prop_modifier,     // модификатор
   double  prop_value         // устанавливаемое значение
   )
```

Параметры

prop_id

[in] Идентификатор свойства индикатора. Значение может быть одним из значений перечисления ENUM_CUSTOMIND_PROPERTY_DOUBLE.

prop_modifier

[in] Модификатор указанного свойства. Только свойства уровней требуют модификатора. Нумерация уровней идет с 0, то есть для задания свойства второму уровню нужно указать единицу (на 1 меньше, чем при использовании директивы компилятора).

prop_value

[in] Значение свойства.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

Примечание

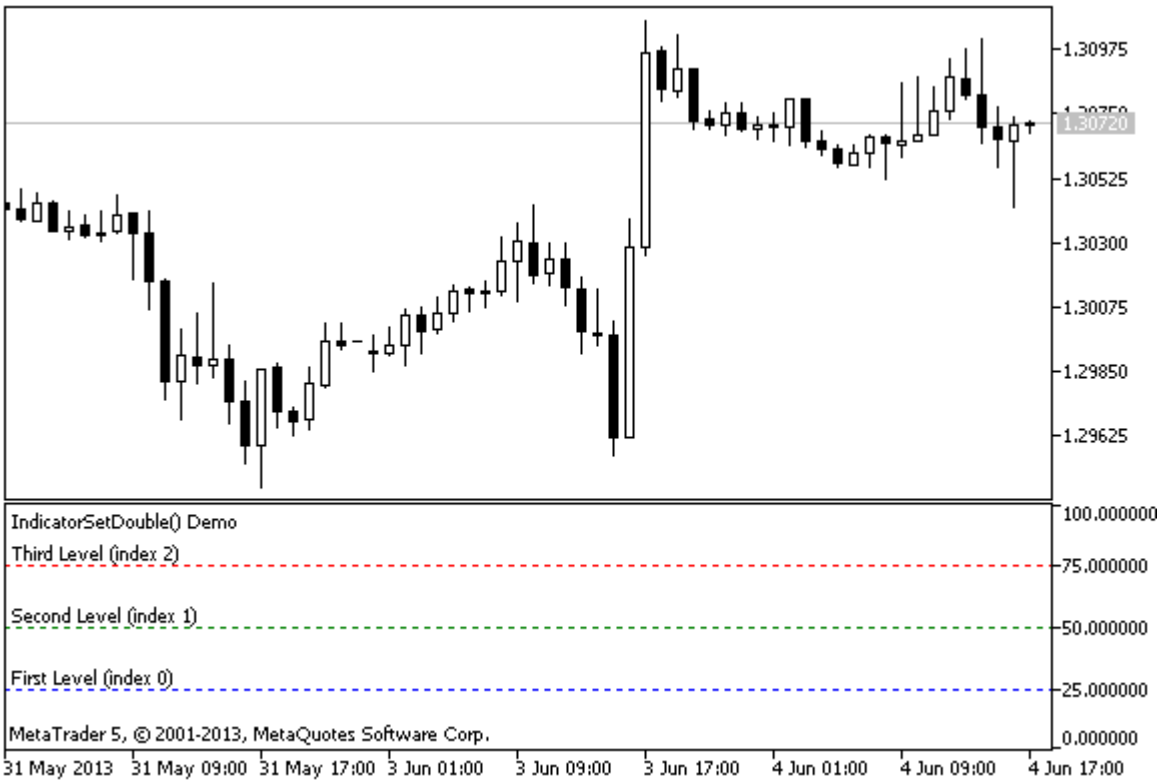
Нумерация свойств (модификаторов) при использовании директивы #property начинается с 1 (единицы), в то время как функция использует нумерацию с 0 (нуля). При неправильном задании номера уровня отображение индикатора может отличаться от того, которое предполагается.

Например, задать значение первого уровня для индикатора в отдельном подокне можно двумя способами:

- property indicator_level**1** 50 - используется 1 для указания номера уровня,
- IndicatorSetDouble(INDICATOR_LEVELVALUE, **0**, 50) - используется 0 для указания первого уровня.

Пример: индикатор-"перевертыш", меняющий максимальное и минимальное значения окна индикатора, а также значения уровней, на которых расположены горизонтальные линии.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_separate_window
//--- установим максимальное и минимальное значения для окна индикатора
#property indicator_minimum  0
#property indicator_maximum  100
//--- зададим показ трех горизонтальных уровней в отдельном окне индикатора
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- установим толщину горизонтальных уровней
#property indicator_levelwidth 1
//--- установим стиль горизонтальных уровней
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- зададим описания горизонтальных уровней
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- зададим короткое имя индикатора
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- зададим каждому уровню свой цвет
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики
   tick_counter++;
//--- на каждом 10-м тике делаем переворот
   if(tick_counter%10==0)
     {
      //--- перевернем знак для значений уровня
      level1=-level1;
      level2=-level2;
      level3=-level3;
      //--- перевернем знак для значений максимума и минимума
      max-=shift;
      min-=shift;
      //--- перевернем значение сдвига
      shift=-shift;
      //--- зададим новые значения уровней
      IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
      //--- зададим новые значения максимума и минимума для окна индикатора
      Print("Set up max = ",max,",   min = ",min);
      IndicatorSetDouble(INDICATOR_MAXIMUM,max);
      IndicatorSetDouble(INDICATOR_MINIMUM,min);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

Стили индикаторов в примерах, Связь между свойствами индикатора и функциями, Стили рисования

© 2000-2025, MetaQuotes Ltd.

IndicatorSetInteger

Задает значение соответствующего свойства индикатора. Свойство индикатора должно быть типа int или color. Существует 2 варианта функции.

Вызов с указанием идентификатора свойства.

```
bool  IndicatorSetInteger(
   int  prop_id,           // идентификатор
   int  prop_value         // устанавливаемое значение
   );
```

Вызов с указанием идентификатора и модификатора свойства.

```
bool  IndicatorSetInteger(
   int  prop_id,           // идентификатор
   int  prop_modifier,     // модификатор
   int  prop_value         // устанавливаемое значение
   )
```

Параметры

prop_id

[in] Идентификатор свойства индикатора. Значение может быть одним из значений перечисления ENUM_CUSTOMIND_PROPERTY_INTEGER.

prop_modifier

[in] Модификатор указанного свойства. Только свойства уровней требуют модификатора.

prop_value

[in] Значение свойства.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

Примечание

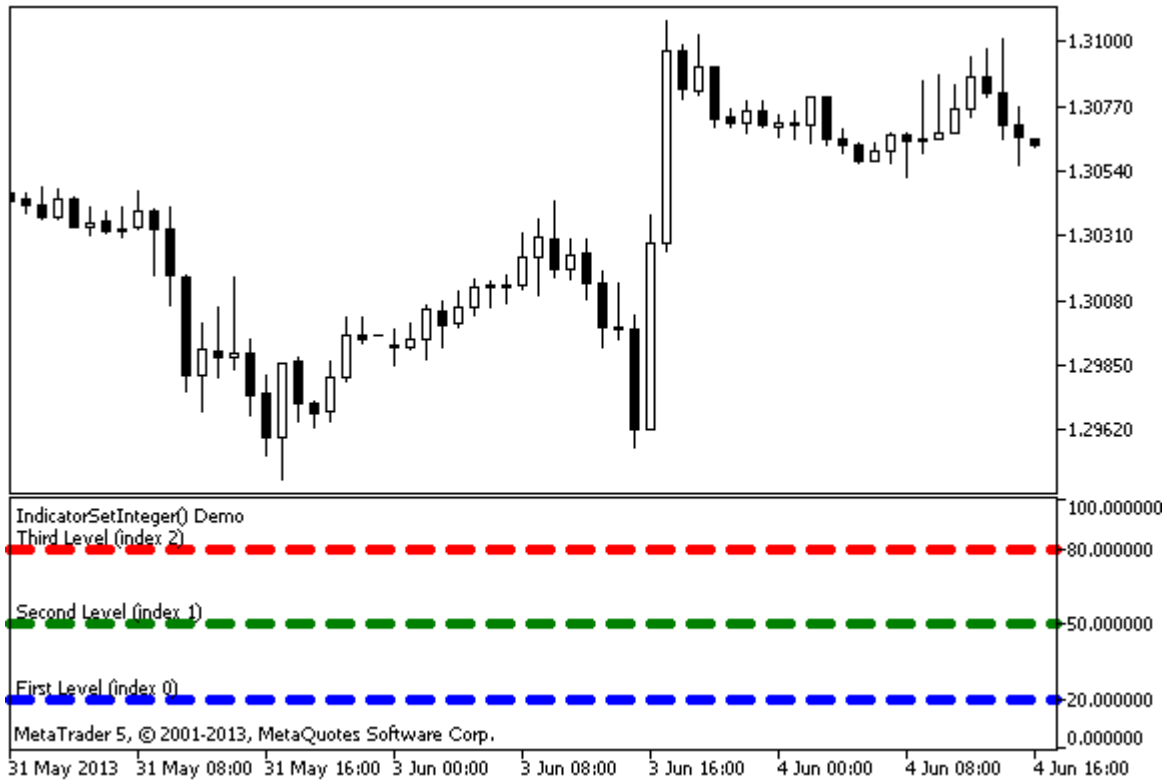
Нумерация свойств (модификаторов) при использовании директивы #property начинается с 1 (единицы), в то время как функция использует нумерацию с 0 (нуля). При неправильном задании номера уровня отображение индикатора может отличаться от того, которое предполагается.

Например, чтобы задать толщину линии первого горизонтального уровня используйте нулевой индекс:

- IndicatorSetInteger(INDICATOR_LEVELWIDTH, **0**, 5) - используется индекс 0 для задания толщины линии первого уровня.

Пример: индикатор, задающий цвет, стиль и толщину горизонтальных уровней индикатора.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- зададим показ трех горизонтальных уровней в отдельном окне индикатора
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//--- установим толщину горизонтальных уровней
#property indicator_levelwidth 5
//--- установим цвет горизонтальных уровней
#property indicator_levelcolor clrAliceBlue
//--- установим стиль горизонтальных уровней
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- зададим описания горизонтальных уровней
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- зададим короткое имя индикатора
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- считаем тики
   tick_counter++;
//--- и устанавливаем цвета горизонтальных уровней в зависимости от счетчика тиков
   ChangeLevelColor(0,tick_counter,3,6,10); // три последних параметра переключают цве
   ChangeLevelColor(1,tick_counter,3,6,8);
   ChangeLevelColor(2,tick_counter,4,7,9);
//--- поменяем стили горизонтальных уровней
   ChangeLevelStyle(0,tick_counter);
   ChangeLevelStyle(1,tick_counter+5);
   ChangeLevelStyle(2,tick_counter+15);
//--- получим толщину линии как остаток целочисленного деления числа тиков на 5
   int width=tick_counter%5;
//--- пройдем по всем горизонтальным уровням и выставим
   for(int l=0;l<3;l++)
      IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Установим цвет горизонтальной линии в отдельном окне индикатора  |
//+------------------------------------------------------------------+
void ChangeLevelColor(int level,      // номер горизонтальной линии
                      int tick_number,// делимое, число для получения остатка деления
                      int f_trigger,  // первый делитель переключения цвета
                      int s_trigger,  // второй делитель переключения цвета
                      int t_trigger)  // третий делитель переключения цвета
  {
   static color colors[3]={clrRed,clrBlue,clrGreen};
//--- индекс цвета из массива colors[]
   int index=-1;
//--- вычислим номер цвета из массива colors[] для раскрашивания горизонтальной линии
   if(tick_number%f_trigger==0)
      index=0;   // если число tick_number делится без остатка на f_trigger
   if(tick_number%s_trigger==0)
      index=1;   // если число tick_number делится без остатка на s_trigger
```

© 2000-2025, MetaQuotes Ltd.

```
   if(tick_number%t_trigger==0)
      index=2;    // если число tick_number делится без остатка на t_trigger
//--- если цвет определен, установим его
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
//---
  }
//+------------------------------------------------------------------+
//| Установим стиль горизонтальной линии в отдельном окне индикатора |
//+------------------------------------------------------------------+
void ChangeLevelStyle(int level,     // номер горизонтальной линии
                      int tick_number// число для получения остатка от деления
                      )
  {
//--- массив для хранения стилей
   static ENUM_LINE_STYLE styles[5]=
     {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//--- индекс стиля из массива styles[]
   int index=-1;
//--- вычислим номер из массива styles[] для установки стиля горизонтальной линии
   if(tick_number%50==0)
      index=5;    // если tick_number делится без остатка на 50, то стиль STYLE_DASHDOT
   if(tick_number%40==0)
      index=4;    // ... стиль 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
//--- если стиль определен, установим его
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
  }
```

Смотри также

Свойства пользовательских индикаторов, Свойства программ (#property), Стили рисования

© 2000-2025, MetaQuotes Ltd.

IndicatorSetString

Задает значение соответствующего свойства индикатора. Свойство индикатора должно быть типа string. Существует 2 варианта функции.

Вызов с указанием идентификатора свойства.

```
bool  IndicatorSetString(
   int     prop_id,           // идентификатор
   string  prop_value         // устанавливаемое значение
   );
```

Вызов с указанием идентификатора и модификатора свойства.

```
bool  IndicatorSetString(
   int     prop_id,           // идентификатор
   int     prop_modifier,     // модификатор
   string  prop_value         // устанавливаемое значение
   )
```

Параметры

prop_id

[in] Идентификатор свойства индикатора. Значение может быть одним из значений перечисления ENUM_CUSTOMIND_PROPERTY_STRING.

prop_modifier

[in] Модификатор указанного свойства. Только свойства уровней требуют модификатора.

prop_value

[in] Значение свойства.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

Примечание

Нумерация свойств (модификаторов) при использовании директивы #property начинается с 1 (единицы), в то время как функция использует нумерацию с 0 (нуля). При неправильном задании номера уровня отображение индикатора может отличаться от того, которое предполагается.

Например, чтобы задать описание первого горизонтального уровня, используйте нулевой индекс:

- IndicatorSetString(INDICATOR_LEVELTEXT, **0**, "First Level") - используется индекс 0 для задания текстового описания первого уровня.

Пример: индикатор, устанавливающий подписи к горизонтальным уровням индикатора.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- зададим показ трех горизонтальных уровней в отдельном окне индикатора
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//--- установим цвет горизонтальных уровней
#property indicator_levelcolor clrRed
//--- установим стиль горизонтальных уровней
#property indicator_levelstyle STYLE_SOLID
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- зададим описания горизонтальных уровней
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- зададим короткое имя индикатора
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
```

© 2000-2025, 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[])
  {
//---

//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Смотри также

Свойства пользовательских индикаторов, Свойства программ (#property)

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetDouble

Задает значение соответствующего свойства соответствующей линии индикатора. Свойство индикатора должно быть типа double.

```
bool  PlotIndexSetDouble(
   int     plot_index,     // индекс графического стиля
   int     prop_id,        // идентификатор свойства
   double  prop_value      // устанавливаемое значение
   );
```

Параметры

plot_index

[in] Индекс графического построения

prop_id

[in] Идентификатор свойства индикатора. Значение может быть одним из значений перечисления ENUM_PLOT_PROPERTY_DOUBLE.

prop_value

[in] Значение свойства.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetInteger

Задает значение соответствующего свойства соответствующей линии индикатора. Свойство индикатора должно быть типа int, char, bool или color. Существует 2 варианта функции.

Вызов с указанием идентификатора свойства.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // индекс графического стиля
   int  prop_id,            // идентификатор свойства
   int  prop_value          // устанавливаемое значение
   );
```

Вызов с указанием идентификатора и модификатора свойства.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // индекс графического стиля
   int  prop_id,            // идентификатор свойства
   int  prop_modifier,      // модификатор свойства
   int  prop_value          // устанавливаемое значение
   )
```

Параметры

plot_index

[in] Индекс графического построения

prop_id

[in] Идентификатор свойства индикатора. Значение может быть одним из значений перечисления ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Модификатор указанного свойства. Только свойства индексов цветов требуют модификатора.

prop_value

[in] Значение свойства.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

Пример: индикатор, рисующий трехцветную линию. Цветовая схема меняется через каждые 5 тиков.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot 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
//--- indicator buffers
double         ColorLineBuffer[];
double         ColorBuffer[];
int            MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- get MA handle
   MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| get color index                                                  |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int getIndexOfColor(int i)
  {
   int j=i%300;
   if(j<100) return(0);// first index
   if(j<200) return(1);// second index
   return(2); // third index
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- first calculation or number of bars was changed
   if(prev_calculated==0)
     {
      //--- copy values of MA into indicator buffer ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);// copying failed - throw away
      //--- now set line color for every bar
      for(int i=0;i<rates_total;i++)
         ColorBuffer[i]=getIndexOfColor(i);
     }
   else
     {
      //--- copy values of MA into indicator buffer ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);

      ticks++;// ticks counting
      if(ticks>=5)//it's time to change color scheme
        {
         ticks=0; // reset counter
         modified++; // counter of color changes
         if(modified>=3)modified=0;// reset counter
         ResetLastError();
         switch(modified)
           {
```

© 2000-2025, MetaQuotes Ltd.

```
            case 0:// first color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
               Print("Color scheme "+modified);
               break;
            case 1:// second color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
               Print("Color scheme "+modified);
               break;
            default:// third color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
               Print("Color scheme "+modified);
           }
        }
      else
        {
         //--- set start position
         limit=prev_calculated-1;
         //--- now we set line color for every bar
         for(int i=limit;i<rates_total;i++)
            ColorBuffer[i]=getIndexOfColor(i);
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetString

Задает значение соответствующего свойства соответствующей линии индикатора. Свойство индикатора должно быть типа string.

```
bool  PlotIndexSetString(
   int     plot_index,     // индекс графического стиля
   int     prop_id,        // идентификатор свойства
   string  prop_value      // устанавливаемое значение
   );
```

Параметры

plot_index

[in] Индекс графического построения

prop_id

[in] Идентификатор свойства индикатора. Значение может быть одним из значений перечисления ENUM_PLOT_PROPERTY_STRING.

prop_value

[in] Значение свойства.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

PlotIndexGetInteger

Возвращает значение соответствующего свойства соответствующей линии индикатора. Свойство должно быть типов int, color, bool или char. Существует 2 варианта функции.

Вызов с указанием идентификатора свойства.

```
int  PlotIndexGetInteger(
   int  plot_index,         // индекс графического стиля
   int  prop_id,            // идентификатор свойства
   );
```

Вызов с указанием идентификатора и модификатора свойства.

```
int  PlotIndexGetInteger(
   int  plot_index,         // индекс графического стиля
   int  prop_id,            // идентификатор свойства
   int  prop_modifier       // модификатор свойства
   )
```

Параметры

plot_index

[in] Индекс графического построения

prop_id

[in] Идентификатор свойства индикатора. Значение может быть одним из значений перечисления ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Модификатор указанного свойства. Только свойства индексов цветов требуют модификатора.

Примечание

Функция предназначена для извлечения настроек рисования соответствующей линии индикатора. Функция работает в паре с функцией PlotIndexSetInteger для копирования свойств рисования из одной линии в другую.

Пример: индикатор, окрашивающий свечи в цвет, зависящий от дня недели. Цвета для каждого дня задаются программным путем.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double          OpenBuffer[];
double          HighBuffer[];
double          LowBuffer[];
double          CloseBuffer[];
double          ColorCandlesColors[];
color           ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                               clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   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);
//--- set number of colors in color buffer
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- set colors for color buffer
   for(int i=1;i<6;i++)
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- set accuracy
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
   printf("We have %u colors of days",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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];
      //--- set color for every candle
      TimeToStruct(time[i],t);
      ColorCandlesColors[i]=t.day_of_week;
      //---
      i++;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

Графические объекты

Группа функций, предназначенных для работы с графическими объектами, относящимися к любому указанному графику.

Функции, задающие свойства графических объектов, а также операции создания ObjectCreate() и перемещения ObjectMove() объектов на графике фактически служат для отправки команд графику. При успешном выполнении этих функций команда попадает в общую очередь событий графика. Визуальное изменение свойств графических объектов производится в процессе обработки очереди событий данного графика.

По этой причине не следует ожидать немедленного визуального обновления графических объектов после вызова данных функций. В общем случае обновление графических объектов на чарте производится терминалом автоматически по событиям изменения - поступление новой котировки, изменения размера окна графика и т.д. Для принудительного обновления графических объектов используйте команду на перерисовку графика ChartRedraw().

Функция	Действие
ObjectCreate	Создает объект заданного типа на указанном графике
ObjectName	Возвращает имя объекта соответствующего типа в указанном графике (указанном подокне графика)
ObjectDelete	Удаляет объект с указанным именем с указанного графика (с указанного подокна графика)
ObjectsDeleteAll	Удаляет все объекты указанного типа с указанного графика (с указанного подокна графика)
ObjectFind	Ищет по имени объект с указанным идентификатором
ObjectGetTimeByValue	Возвращает значение времени для указанного значения цены объекта
ObjectGetValueByTime	Возвращает ценовое значение объекта для указанного времени
ObjectMove	Изменяет координаты указанной точки привязки объекта
ObjectsTotal	Возвращает количество объектов указанного типа в указанном графике (указанном подокне графика)
ObjectGetDouble	Возвращает значение типа double соответствующего свойства объекта
ObjectGetInteger	Возвращает целочисленное значение соответствующего свойства объекта
ObjectGetString	Возвращает значение типа string соответствующего свойства объекта
ObjectSetDouble	Устанавливает значение соответствующего свойства объекта
ObjectSetInteger	Устанавливает значение соответствующего свойства объекта
ObjectSetString	Устанавливает значение соответствующего свойства объекта

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
TextSetFont	Устанавливает шрифт для вывода текста методами рисования (по умолчанию используется шрифт Arial 20)
TextOut	Выводит текст в пользовательский массив (буфер), предназначенный для создания графического ресурса
TextGetSize	Возвращает ширину и высоту строки при текущих настройках шрифта

Каждый графический объект должен иметь имя, уникальное в пределах одного графика, включая его подокна. Изменение имени графического объекта формирует два события: первое - это событие удаления объекта со старым именем, и второе – событие создания графического объекта с новым именем.

После создания объекта или модификации свойств объекта рекомендуется вызывать функцию ChartRedraw(), которая отдает терминалу команду на принудительную отрисовку графика (и всех видимых на нем объектов).

© 2000-2025, MetaQuotes Ltd.

ObjectCreate

Создает объект с указанным именем, типом и начальными координатами в указанном подокне графика. При создании можно указать до 30 координат.

```
bool  ObjectCreate(
   long          chart_id,       // идентификатор графика
   string        name,           // имя объекта
   ENUM_OBJECT   type,           // тип объекта
   int           sub_window,     // индекс окна
   datetime      time1,          // время первой точки привязки
   double        price1,         // цена первой точки привязки
   ...
   datetime      timeN=0,        // время N-ой точки привязки
   double        priceN=0,       // цена N-ой точки привязки
   ...
   datetime      time30=0,       // время 30-й точки привязки
   double        price30=0       // цена 30-точки привязки
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта. Имя должно быть уникальным в пределах одного графика, включая его подокна.

type

[in] Тип объекта. Значение может быть одним из значений перечисления ENUM_OBJECT.

sub_window

[in] Номер подокна графика. 0 означает главное окно графика. Указанное подокно должно существовать, в противном случае функция возвращает false.

time1

[in] ВременнАя координата первой привязки.

price1

[in] Ценовая координата первой точки привязки.

timeN=0

[in] ВременнАя координата N-ой точки привязки.

priceN=0

[in] Ценовая координата N-ой точки привязки.

time30=0

[in] ВременнАя координата тридцатой точки привязки.

price30=0

© 2000-2025, MetaQuotes Ltd.

[in] Ценовая координата тридцатой точки привязки.

Возвращаемое значение

Возвращает true при успешной постановке команды в очередь указанного графика, иначе false. Если объект был уже создан ранее, то производится попытка изменить его координаты.

Примечание

При вызове ObjectCreate() всегда используется асинхронный вызов, поэтому функция возвращает только результат постановки команды в очередь графика. В этом случае true означает только то, что команда успешно поставлена в очередь, сам результат её выполнения неизвестен.

Для проверки результата выполнения можно использовать функцию ObjectFind() или любые функции, запрашивающие свойства объекта, например вида ObjectGetXXX. Но при этом следует иметь в виду, что такие функции ставятся в конец очереди команд графика и дожидаются результата выполнения (так как являются синхронными вызовами), то есть могут быть затратными по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

Имя графического объекта не должно превышать 63 символа.

Нумерация подокон графика (если на графике есть подокна с индикаторами) начинается с 1. Главное окно графика есть всегда и имеет индекс 0.

Большое количество точек привязки (до 30-ти) предусмотрено для будущего использования. В то же время ограничение только 30-тью возможными точками привязки для графических объектов обусловлено тем, что при вызове функции количество параметров не должно превышать 64.

При переименовании графического объекта одновременно формируются два события, которые можно обработать в эксперте или индикаторе функцией OnChartEvent():

- событие удаления объекта со старым именем;
- событие создания графического объекта с новым именем.

Для создания каждого из типов объектов требуется задать определенное количество точек привязки:

Идентификатор	Описание	Точки привязки
OBJ_VLINE	Вертикальная линия	Одна точка привязки. Фактически используется только координата по оси времени.
OBJ_HLINE	Горизонтальная линия	Одна точка привязки. Фактически используется только координата по оси цены.
OBJ_TREND	Трендовая линия	Две точки привязки.
OBJ_TRENDBYANGLE	Трендовая линия по углу	Две точки привязки.
OBJ_CYCLES	Циклические линии	Две точки привязки.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Точки привязки
OBJ_ARROWED_LINE	Объект "Линия со стрелкой"	Две точки привязки.
OBJ_CHANNEL	Равноудаленный канал	Три точки привязки.
OBJ_STDDEVCHANNEL	Канал стандартного отклонения	Две точки привязки.
OBJ_REGRESSION	Канал на линейной регрессии	Две точки привязки.
OBJ_PITCHFORK	Вилы Эндрюса	Три точки привязки.
OBJ_GANNLINE	Линия Ганна	Две точки привязки.
OBJ_GANNFAN	Веер Ганна	Две точки привязки.
OBJ_GANNGRID	Сетка Ганна	Две точки привязки.
OBJ_FIBO	Уровни Фибоначчи	Две точки привязки.
OBJ_FIBOTIMES	Временные зоны Фибоначчи	Две точки привязки.
OBJ_FIBOFAN	Веер Фибоначчи	Две точки привязки.
OBJ_FIBOARC	Дуги Фибоначчи	Две точки привязки.
OBJ_FIBOCHANNEL	Канал Фибоначчи	Три точки привязки.
OBJ_EXPANSION	Расширение Фибоначчи	Три точки привязки.
OBJ_ELLIOTWAVE5	5-волновка Эллиота	Пять точек привязки.
OBJ_ELLIOTWAVE3	3-волновка Эллиота	Три точки привязки.
OBJ_RECTANGLE	Прямоугольник	Две точки привязки.
OBJ_TRIANGLE	Треугольник	Три точки привязки.
OBJ_ELLIPSE	Эллипс	Три точки привязки.
OBJ_ARROW_THUMB_UP	Знак "Хорошо" (большой палец вверх)	Одна точка привязки.
OBJ_ARROW_THUMB_DOWN	Знак "Плохо" (большой палец вниз)	Одна точка привязки.
OBJ_ARROW_UP	Знак "Стрелка вверх"	Одна точка привязки.
OBJ_ARROW_DOWN	Знак "Стрелка вниз"	Одна точка привязки.
OBJ_ARROW_STOP	Знак "Стоп"	Одна точка привязки.
OBJ_ARROW_CHECK	Знак "Птичка" (галка)	Одна точка привязки.
OBJ_ARROW_LEFT_PRICE	Левая ценовая метка	Одна точка привязки.
OBJ_ARROW_RIGHT_PRICE	Правая ценовая метка	Одна точка привязки.
OBJ_ARROW_BUY	Знак "Buy"	Одна точка привязки.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Точки привязки
OBJ_ARROW_SELL	Знак "Sell"	Одна точка привязки.
OBJ_ARROW	Объект "Стрелка"	Одна точка привязки.
OBJ_TEXT	Объект "Текст"	Одна точка привязки.
OBJ_LABEL	Объект "Текстовая метка"	Положение задается при помощи свойств OBJPROP_XDISTANCE и OBJPROP_YDISTANCE.
OBJ_BUTTON	Объект "Кнопка"	Положение задается при помощи свойств OBJPROP_XDISTANCE и OBJPROP_YDISTANCE.
OBJ_CHART	Объект "График"	Положение задается при помощи свойств OBJPROP_XDISTANCE и OBJPROP_YDISTANCE.
OBJ_BITMAP	Объект "Рисунок"	Одна точка привязки.
OBJ_BITMAP_LABEL	Объект "Графическая метка"	Положение задается при помощи свойств OBJPROP_XDISTANCE и OBJPROP_YDISTANCE.
OBJ_EDIT	Объект "Поле ввода"	Положение задается при помощи свойств OBJPROP_XDISTANCE и OBJPROP_YDISTANCE.
OBJ_EVENT	Объект "Событие", соответствующий событию в экономическом календаре	Одна точка привязки. Фактически используется только координата по оси времени.
OBJ_RECTANGLE_LABEL	Объект "Прямоугольная метка"для создания и оформления пользовательского графического интерфейса.	Положение задается при помощи свойств OBJPROP_XDISTANCE и OBJPROP_YDISTANCE.

© 2000-2025, MetaQuotes Ltd.

ObjectName

Возвращает имя соответствующего объекта в указанном чарте, указанном подокне указанного чарта, указанного типа.

```
string  ObjectName(
   long  chart_id,           // идентификатор графика
   int   pos,                // номер в списке объектов
   int   sub_window=-1,      // номер окна
   int   type=-1             // тип объекта
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

pos

[in] Порядковый номер объекта согласно указанного фильтра по номеру подокна и типу.

sub_window=-1

[in] Номер подокна графика. 0 означает главное окно графика, -1 означает все подокна графика, включая главное окно.

type=-1

[in] Тип объекта. Значение может быть одним из значений перечисления ENUM_OBJECT. -1 означает все типы.

Возвращаемое значение

Имя объекта в случае успеха.

Примечание

Функция использует синхронный вызов - это означает, что функция дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом, и поэтому данная функция может быть затратной по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

При переименовании графического объекта одновременно формируются два события, которые можно обработать в эксперте или индикаторе функцией OnChartEvent():

- событие удаления объекта со старым именем;
- событие создания графического объекта с новым именем.

© 2000-2025, MetaQuotes Ltd.

ObjectDelete

Удаляет объект с указанным именем с указанного графика.

```
bool  ObjectDelete(
   long    chart_id,      // идентификатор графика
   string  name           // имя объекта
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя удаляемого объекта.

Возвращаемое значение

Возвращает true при успешной постановке команды в очередь указанного графика, иначе false.

Примечание

При вызове ObjectDelete() всегда используется асинхронный вызов, поэтому функция возвращает только результат постановки команды в очередь графика. В этом случае true означает только то, что команда успешно поставлена в очередь, сам результат её выполнения неизвестен.

Для проверки результата выполнения можно использовать функцию ObjectFind() или любые функции, запрашивающие свойства объекта, например вида ObjectGetXXX. Но при этом следует иметь в виду, что такие функции ставятся в конец очереди команд графика и дожидаются результата выполнения (так как являются синхронными вызовами), то есть могут быть затратными по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

При переименовании графического объекта одновременно формируются два события, которые можно обработать в эксперте или индикаторе функцией OnChartEvent():

- событие удаления объекта со старым именем;
- событие создания графического объекта с новым именем.

© 2000-2025, MetaQuotes Ltd.

ObjectsDeleteAll

Удаляет все объекты в указанном графике, указанном подокне указанного графика, указанного типа. Существует два варианта функции:

```
int  ObjectsDeleteAll(
   long  chart_id,            // идентификатор графика
   int   sub_window=-1,       // индекс окна
   int   type=-1              // тип объекта для удаления
   );
```

Удаляет по префиксу имени в подокне графика все объекты указанного типа.

```
int  ObjectsDeleteAll(
   long           chart_id,   // идентификатор графика
   const string     prefix,   // префикс имени объекта
   int        sub_window=-1,  // индекс окна
   int      object_type=-1    // тип объекта для удаления
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

prefix

[in] Префикс, по которому будут удалены все объекты, чьи имена начинаются с данного набора символов. Префикс можно указывать как 'name' или 'name*' - оба варианта работают одинаково. Если в качестве префикса указана пустая строка, то будут удалены объекты с любым именем.

sub_window=-1

[in] Номер подокна графика. 0 означает главное окно графика, -1 означает все подокна графика, включая главное окно.

type=-1

[in] Тип объекта. Значение может быть одним из значений перечисления ENUM_OBJECT. -1 означает все типы.

Возвращаемое значение

Возвращает количество удаленных объектов. Для получения дополнительной информации об ошибке необходимо вызвать функцию функцию GetLastError().

Примечание

Функция использует синхронный вызов - это означает, что функция дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом, и поэтому данная функция может быть затратной по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

© 2000-2025, MetaQuotes Ltd.

ObjectFind

Ищет объект с указанным именем на графике с указанным идентификатором.

```
int  ObjectFind(
   long    chart_id,      // идентификатор графика
   string  name           // имя объекта
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя искомого объекта.

Возвращаемое значение

В случае удачи функция возвращает номер подокна (0 означает главное окно графика), в котором находится найденный объект. Если объект не найден, то функция возвращает отрицательное число. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция использует синхронный вызов - это означает, что функция дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом, и поэтому данная функция может быть затратной по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

При переименовании графического объекта одновременно формируются два события, которые можно обработать в эксперте или индикаторе функцией OnChartEvent():

- событие удаления объекта со старым именем;
- событие создания графического объекта с новым именем.

© 2000-2025, MetaQuotes Ltd.

ObjectGetTimeByValue

Возвращает значение времени для указанного значения цены указанного объекта.

```
datetime  ObjectGetTimeByValue(
   long     chart_id,      // идентификатор графика
   string   name,          // имя объекта
   double   value,         // цена
   int      line_id        // номер линии
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта.

value

[in] Значение цены.

line_id

[in] Идентификатор линии.

Возвращаемое значение

Значение времени для указанного значения цены указанного объекта.

Примечание

Функция использует синхронный вызов - это означает, что функция дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом, и поэтому данная функция может быть затратной по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

Так как объект в одной координате цены может иметь несколько значений, то необходимо указать номер линии. Эта функция применима только для следующих объектов:

- Трендовая линия (OBJ_TREND)
- Трендовая линия по углу (OBJ_TRENDBYANGLE)
- Линия Ганна (OBJ_GANNLINE)
- Равноудаленный канал (OBJ_CHANNEL) - 2 линии
- Канал на линейной регрессии (OBJ_REGRESSION) - 3 линии
- Канал стандартного отклонения (OBJ_STDDEVCHANNEL) - 3 линии
- Линия со стрелкой (OBJ_ARROWED_LINE)

Смотри также

Типы объектов

© 2000-2025, MetaQuotes Ltd.

ObjectGetValueByTime

Возвращает значение цены для указанного времени указанного объекта.

```
double  ObjectGetValueByTime(
   long      chart_id,     // идентификатор графика
   string    name,         // имя объекта
   datetime  time,         // время
   int       line_id       // номер линии
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта.

time

[in] Значение времени.

line_id

[in] Идентификатор линии.

Возвращаемое значение

Значение цены для указанного времени указанного объекта.

Примечание

Функция использует синхронный вызов - это означает, что функция дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом, и поэтому данная функция может быть затратной по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

Так как объект в одной координате цены может иметь несколько значений, то необходимо указать номер линии. Эта функция применима только для следующих объектов:

- Трендовая линия (OBJ_TREND)
- Трендовая линия по углу (OBJ_TRENDBYANGLE)
- Линия Ганна (OBJ_GANNLINE)
- Равноудаленный канал (OBJ_CHANNEL) - 2 линии
- Канал на линейной регрессии (OBJ_REGRESSION) - 3 линии
- Канал стандартного отклонения (OBJ_STDDEVCHANNEL) - 3 линии
- Линия со стрелкой (OBJ_ARROWED_LINE)

Смотри также

Типы объектов

© 2000-2025, MetaQuotes Ltd.

ObjectMove

Изменяет координаты указанной точки привязки объекта.

```
bool  ObjectMove(
   long      chart_id,         // идентификатор графика
   string    name,             // имя объекта
   int       point_index,      // номер привязки
   datetime  time,             // время
   double    price             // цена
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта.

point_index

[in] Номер точки привязки. Количество точек привязки зависит от типа объекта.

time

[in] ВременнАя координата указанной точки привязки.

price

[in] Ценовая координата указанной точки привязки.

Возвращаемое значение

Возвращает true при успешной постановке команды в очередь указанного графика, иначе false.

Примечание

При вызове ObjectMove() всегда используется асинхронный вызов, поэтому функция возвращает только результат постановки команды в очередь графика. В этом случае true означает только то, что команда успешно поставлена в очередь, сам результат её выполнения неизвестен.

Для проверки результата выполнения можно использовать функцию, запрашивающую свойства объекта, например вида ObjectGetXXX. Но при этом следует иметь в виду, что такие функции ставятся в конец очереди команд графика и дожидаются результата выполнения (так как являются синхронными вызовами), то есть могут быть затратными по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

© 2000-2025, MetaQuotes Ltd.

ObjectsTotal

Возвращает количество объектов в указанном чарте, указанном подокне указанного чарта, указанного типа.

```
int  ObjectsTotal(
   long  chart_id,           // идентификатор графика
   int   sub_window=-1,      // индекс окна
   int   type=-1             // тип объекта
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

nwin=-1

[in] Номер подокна графика. 0 означает главное окно графика, -1 означает все подокна графика, включая главное окно.

type=-1

[in] Тип объекта. Значение может быть одним из значений перечисления ENUM_OBJECT. -1 означает все типы.

Возвращаемое значение

Количество объектов.

Примечание

Функция использует синхронный вызов - это означает, что функция дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом, и поэтому данная функция может быть затратной по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

© 2000-2025, MetaQuotes Ltd.

ObjectSetDouble

Задает значение соответствующего свойства объекта. Свойство объекта должно быть типа double. Существует 2 варианта функции.

Установка значения свойства, не имеющего модификатора

```
bool  ObjectSetDouble(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // свойство
   double                        prop_value         // значение
   );
```

Установка значения свойства с указанием модификатора

```
bool  ObjectSetDouble(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // свойство
   int                           prop_modifier,     // модификатор
   double                        prop_value         // значение
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта.

prop_id

[in] Идентификатор свойства объекта. Значение может быть одним из значений перечисления ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Модификатор указанного свойства. Означает номер уровня в инструментах Фибоначчи и в графическом объекте Вилы Эндрюса. Нумерация уровней начинается с нуля.

prop_value

[in] Значение свойства.

Возвращаемое значение

Возвращает true только в том случае, если команда на изменение свойств графического объекта успешно отправлена графику, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

© 2000-2025, MetaQuotes Ltd.

Функция использует асинхронный вызов - это означает, что функция не дожидается выполнения команды, успешно поставленной в очередь указанного графика, а сразу же возвращает управление.

Для проверки результата выполнения на чужом графике можно использовать функцию, запрашивающую указанное свойство объекта. Но при этом следует иметь в виду, что такие функции ставятся в конец очереди команд чужого графика и дожидаются результата выполнения, то есть могут быть затратными по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

Пример создания Фибо-объекта и добавления нового уровня в нем

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- вспомогательные массивы
   double high[],low[],price1,price2;
   datetime time[],time1,time2;
//--- скопируем цены открытия - 100 последних баров хватит
   int copied=CopyHigh(Symbol(),0,0,100,high);
   if(copied<=0)
     {
      Print("Не удалось скопировать значения ценовой серии High");
      return;
     }
//--- скопируем цены закрытия - 100 последних баров хватит
   copied=CopyLow(Symbol(),0,0,100,low);
   if(copied<=0)
     {
      Print("Не удалось скопировать значения ценовой серии Low");
      return;
     }
//--- скопируем время открытия для 100 последних баров
   copied=CopyTime(Symbol(),0,0,100,time);
   if(copied<=0)
     {
      Print("Не удалось скопировать значения ценовой серии Time");
      return;
     }
//--- организуем доступ к скопированным данным как к таймсериям - задом наперед
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(time,true);

//--- координаты первой точки привязки Фибо-объекта
   price1=high[70];
   time1=time[70];
//--- координаты первой точки привязки Фибо-объекта
```

© 2000-2025, MetaQuotes Ltd.

```
   price2=low[50];
   time2=time[50];

//--- пора создать и сам Фибо-объект
   bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
   if(created) // если объект создан удачно
     {
      //--- установим цвет Фибо0уровней
      ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
      //--- кстати, а сколько у нас Фибо-уровней ?
      int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
      Print("Fibo levels before = ",levels);
      //---выведем в Журнал=> номер уровня:значения описание_уровня
      for(int i=0;i<levels;i++)
        {
         Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
               "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
      //--- попробуем увеличить количество уровней на единицу
      bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
      if(!modified) // не удалось количество уровней изменить
        {
         Print("Не удалось изменить количество уровней в Фибо, ошибка ",GetLastError());
        }
      //--- просто сообщим
      Print("Fibo levels after = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
      //--- зададим значение для вновь созданного уровня
      bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
      if(added) // удалось задать значение для уровня
        {
         Print("Удалось установить еще один уровень Fibo");
         //--- также не забыть задать описание уровня
         ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
         ChartRedraw(0);
         //--- получим актуальное значения количества уровней в Фибо-объекте
         levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
         Print("Fibo levels after adding = ",levels);
         //--- еще раз выведем все уровни - просто чтобы убедиться
         for(int i=0;i<levels;i++)
           {
            Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                  "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
           }

        }
      else // неудача при попытке увеличить количество уровней в Фибо-объекте
        {
         Print("Не удалось установить ещеь один уровень Fibo. Ошибка ",GetLastError());
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
```

Смотри также

Типы объектов, Свойства объектов

© 2000-2025, MetaQuotes Ltd.

ObjectSetInteger

Задает значение соответствующего свойства объекта. Свойство объекта должно быть типов datetime, int, color, bool или char. Существует 2 варианта функции.

Установка значения свойства, не имеющего модификатора

```
bool  ObjectSetInteger(
   long                          chart_id,      // идентификатор графика
   string                        name,          // имя
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // свойство
   long                          prop_value     // значение
   );
```

Установка значения свойства с указанием модификатора

```
bool  ObjectSetInteger(
   long                          chart_id,      // идентификатор графика
   string                        name,          // имя
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // свойство
   int                           prop_modifier, // модификатор
   long                          prop_value     // значение
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта.

prop_id

[in] Идентификатор свойства объекта. Значение может быть одним из значений перечисления ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Модификатор указанного свойства. Означает номер уровня в инструментах Фибоначчи и в графическом объекте Вилы Эндрюса. Нумерация уровней начинается с нуля.

prop_value

[in] Значение свойства.

Возвращаемое значение

Возвращает true только в том случае, если команда на изменение свойств графического объекта успешно отправлена графику, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

© 2000-2025, MetaQuotes Ltd.

Функция использует асинхронный вызов - это означает, что функция не дожидается выполнения команды, успешно поставленной в очередь указанного графика, а сразу же возвращает управление.

Для проверки результата выполнения на чужом графике можно использовать функцию, запрашивающую указанное свойство объекта. Но при этом следует иметь в виду, что такие функции ставятся в конец очереди команд чужого графика и дожидаются результата выполнения, то есть могут быть затратными по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

Пример создания таблицы Web-цветов

```
//+------------------------------------------------------------------+
//|                                                Table of Web Colors|
//|                        Copyright 2011, MetaQuotes Software Corp |
//|                                             https://www.metaquotes.net |
//+------------------------------------------------------------------+
#define X_SIZE 140       // ширина объекта OBJ_EDIT
#define Y_SIZE 33        // ширина объекта OBJ_EDIT
//+------------------------------------------------------------------+
//| Массив Web-цветов
//+------------------------------------------------------------------+
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,clrC
   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
  };
//+------------------------------------------------------------------+
//| Создание и инициализация объекта OBJ_EDIT                        |
//+------------------------------------------------------------------+
void CreateColorBox(int x,int y,color c)
  {
//--- сгенерируем по имени цвета имя для нового объекта
      string name="ColorBox_"+(string)x+"_"+(string)y;
//--- создадим новый объект OBJ_EDIT
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
     {
      Print("Не удалось создать объект: '",name,"'");
      return;
     }
//--- зададим координаты точки привязки, ширину и высоту в пикселях
   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);
//--- установим цвет текста для объекта
   if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
   else            ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
//--- установим цвет фона
   ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
//--- установим текст объекта OBJ_EDIT соответствующим цвету фона
   ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
  }
//+------------------------------------------------------------------+
//| Функция запуска скрипта на выполнение                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим таблицу из цветовых блоков 7x20
   for(uint i=0;i<140;i++)
      CreateColorBox(i%7,i/7,ExtClr[i]);
  }
```

Смотри также

Типы объектов, Свойства объектов

© 2000-2025, MetaQuotes Ltd.

ObjectSetString

Задает значение соответствующего свойства объекта. Свойство объекта должно быть типа string. Существует 2 варианта функции.

Установка значения свойства, не имеющего модификатора

```
bool  ObjectSetString(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // свойство
   string                        prop_value         // значение
   );
```

Установка значения свойства с указанием модификатора

```
bool  ObjectSetString(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // свойство
   int                           prop_modifier,     // модификатор
   string                        prop_value         // значение
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта.

prop_id

[in] Идентификатор свойства объекта. Значение может быть одним из значений перечисления ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Модификатор указанного свойства. Означает номер уровня в инструментах Фибоначчи и в графическом объекте Вилы Эндрюса. Нумерация уровней начинается с нуля.

prop_value

[in] Значение свойства.

Возвращаемое значение

Возвращает true только в том случае, если команда на изменение свойств графического объекта успешно отправлена графику, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

© 2000-2025, MetaQuotes Ltd.

Функция использует асинхронный вызов – это означает, что функция не дожидается выполнения команды, успешно поставленной в очередь указанного графика, а сразу же возвращает управление.

Для проверки результата выполнения на чужом графике можно использовать функцию, запрашивающую указанное свойство объекта. Но при этом следует иметь в виду, что такие функции ставятся в конец очереди команд чужого графика и дожидаются результата выполнения, то есть могут быть затратными по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

При переименовании графического объекта одновременно формируются два события, которые можно обработать в эксперте или индикаторе функцией OnChartEvent():

- событие удаления объекта со старым именем;
- событие создания графического объекта с новым именем.

© 2000-2025, MetaQuotes Ltd.

ObjectGetDouble

Возвращает значение соответствующего свойства объекта. Свойство объекта должно быть типа double. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
double  ObjectGetDouble(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя объекта
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // идентификатор свойства
   int                           prop_modifier=0    // модификатор свойства, если тр
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  ObjectGetDouble(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя объекта
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // идентификатор свойства
   int                           prop_modifier,     // модификатор свойства
   double&                       double_var         // сюда примем значение свойства
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта.

prop_id

[in] Идентификатор свойства объекта. Значение может быть одним из значений перечисления ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Модификатор указанного свойства. Для первого варианта по умолчанию значение модификатора равно 0. Большинство свойств не требуют модификатора. Означает номер уровня в инструментах Фибоначчи и в графическом объекте Вилы Эндрюса. Нумерация уровней начинается с нуля.

double_var

[out] Переменная типа double, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа double для первого варианта вызова.

© 2000-2025, MetaQuotes Ltd.

Для второго варианта вызова возвращает true, если данное свойство поддерживается и значение было помещено в переменную double_var, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция использует синхронный вызов – это означает, что функция дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом, и поэтому данная функция может быть затратной по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

© 2000-2025, MetaQuotes Ltd.

ObjectGetInteger

Возвращает значение соответствующего свойства объекта. Свойство объекта должно быть типов datetime, int, color, bool или char. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
long  ObjectGetInteger(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя объекта
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // идентификатор свойства
   int                           prop_modifier=0    // модификатор свойства, если
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  ObjectGetInteger(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя объекта
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // идентификатор свойства
   int                           prop_modifier,     // модификатор свойства
   long&                         long_var           // сюда примем значение свойст
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта.

prop_id

[in] Идентификатор свойства объекта. Значение может быть одним из значений перечисления ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Модификатор указанного свойства. Для первого варианта по умолчанию значение модификатора равно 0. Большинство свойств не требуют модификатора. Означает номер уровня в инструментах Фибоначчи и в графическом объекте Вилы Эндрюса. Нумерация уровней начинается с нуля.

long_var

[out] Переменная типа long, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа long для первого варианта вызова.

© 2000-2025, MetaQuotes Ltd.

Для второго варианта вызова возвращает true, если данное свойство поддерживается и значение было помещено в переменную long_var, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция использует синхронный вызов - это означает, что функция дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом, и поэтому данная функция может быть затратной по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

© 2000-2025, MetaQuotes Ltd.

ObjectGetString

Возвращает значение соответствующего свойства объекта. Свойство объекта должно быть типа string. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
string  ObjectGetString(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя объекта
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // идентификатор свойства
   int                           prop_modifier=0    // модификатор свойства, если тр
   );
```

2. Возвращает true или false в зависимости от успешности выполнения функции. В случае успеха значение свойства помещается в приемную переменную, передаваемую по ссылке последним параметром.

```
bool  ObjectGetString(
   long                          chart_id,          // идентификатор графика
   string                        name,              // имя объекта
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // идентификатор свойства
   int                           prop_modifier,     // модификатор свойства
   string&                       string_var         // сюда примем значение свойства
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

name

[in] Имя объекта.

prop_id

[in] Идентификатор свойства объекта. Значение может быть одним из значений перечисления ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Модификатор указанного свойства. Для первого варианта по умолчанию значение модификатора равно 0. Большинство свойств не требуют модификатора. Означает номер уровня в инструментах Фибоначчи и в графическом объекте Вилы Эндрюса. Нумерация уровней начинается с нуля.

string_var

[out] Переменная типа string, принимающая значение запрашиваемого свойства.

Возвращаемое значение

Значение типа string для первого варианта вызова.

© 2000-2025, MetaQuotes Ltd.

Для второго варианта вызова возвращает true, если данное свойство поддерживается и значение было помещено в переменную string_var, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция использует синхронный вызов – это означает, что функция дожидается выполнения всех команд, которые были помещены в очередь графика перед её вызовом, и поэтому данная функция может быть затратной по времени. Нужно иметь это обстоятельство в виду, если ведется работа с большим количеством объектов на графике.

При переименовании графического объекта одновременно формируются два события, которые можно обработать в эксперте или индикаторе функцией OnChartEvent():

- событие удаления объекта со старым именем;
- событие создания графического объекта с новым именем.

© 2000-2025, MetaQuotes Ltd.

TextSetFont

Устанавливает шрифт для вывода текста методами рисования и возвращает результат успешности этой операции. По умолчанию используется шрифт Arial и размер -120 (12 pt).

```
bool  TextSetFont(
   const string  name,            // имя шрифта или путь к файлу шрифта на диске
   int           size,            // размер шрифта
   uint          flags,           // комбинация флагов
   int           orientation=0    // угол наклона текста
   );
```

Параметры

name

[in] Имя шрифта в системе, или имя ресурса, содержащего шрифт, или путь к файлу шрифта на диске.

size

[in] Размер шрифта, который может задаваться положительными и отрицательными значениями. При положительных значениях размер выводимого текста не зависит от настроек размеров шрифтов в операционной системе. При отрицательных значениях значение задается в десятых долях пункта и размер текста будет зависеть от настроек системы ("стандартный масштаб" или "крупный масштаб"). Более подробно о разнице в режимах смотрите в Примечании.

flags

[in] Комбинация флагов, описывающих стиль шрифт.

orientation

[in] Угол наклона текста по горизонтали к оси X, единица измерения равна 0.1 градуса. То есть *orientation=450* означает наклон в 45 градусов.

Возвращаемое значение

Возвращает true в случае успешной установки текущего шрифта, иначе false. Возможные коды ошибок:

- ERR_INVALID_PARAMETER(4003) - *name* представляет NULL или "" (пустая строка),
- ERR_INTERNAL_ERROR(4001) - ошибка операционной системы (например, попытка создания несуществующего шрифта).

Примечание

Если в имени шрифта используется "::", то шрифт загружается из ресурса EX5. Если имя шрифта *name* указано с расширением, то шрифт загружается из файла, при этом - если путь начинается с "\" или "/", то файл ищется относительно каталога MQL5, иначе ищется относительно пути EX5-файла, вызвавшего функцию TextSetFont().

Размер шрифта задается положительными или отрицательными значениями, знак определяет зависимость размера текста от настроек операционной системы (масштаба шрифта).

- Если размер задается положительным числом, то при отображении логического шрифта в физический происходит преобразование размера в физические единицы измерения устройства (пиксели) и этот размер соответствует высоте ячеек символов из доступных шрифтов. Не

© 2000-2025, MetaQuotes Ltd.

рекомендуется в тех случаях, когда предполагается совместное использование на графике текстов, выведенных функцией TextOut(), и текстов, отображаемых с помощью графического объекта OBJ_LABEL ("Текстовая метка").

- Если размер задается отрицательным числом, то указанный размер предполагается заданным в десятых долях логического пункта (значение -350 равно 35 логических пунктов) и делится на 10, а затем полученное значение преобразуется в физические единицы измерения устройства (пиксели) и соответствует абсолютному значению высоты символа из доступных шрифтов. Чтобы получить на экране текст такого же размера, как и в объекте OBJ_LABEL, возьмите указанный в свойствах объекта размер шрифта и умножьте на -10.

Флаги могут использоваться в виде комбинации флагов стиля с одним из флагов, задающем толщину шрифта. Наименования флагов приведены ниже.

Флаги для задания стиля начертания шрифта

Флаг	Описание
FONT_ITALIC	Курсив
FONT_UNDERLINE	Подчёркивание
FONT_STRIKEOUT	Перечёркивание

Флаги для задания толщины шрифта

Флаг
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

Смотри также

© 2000-2025, MetaQuotes Ltd.

Ресурсы, ResourceCreate(), ResourceSave(), TextOut()

© 2000-2025, MetaQuotes Ltd.

TextOut

Выводит текст в пользовательский массив (буфер) и возвращает результат успешности этой операции. Данный массив предназначается для создания графического ресурса.

```
bool  TextOut(
   const string       text,          // выводимый текст
   int                x,             // координата X
   int                y,             // координата Y
   uint               anchor,        // способ привязки
   uint               &data[],       // буфер для вывода
   uint               width,         // ширина буфера в точках
   uint               height,        // высота буфера в точках
   uint               color,         // цвет текста
   ENUM_COLOR_FORMAT  color_format   // формат цвета для вывода
   );
```

Параметры

text

[in] Выводимый текст, который будет записан в буфер. Осуществляется только однострочный вывод текста.

x

[in] Координата X точки привязки для выводимого текста.

y

[in] Координата Y точки привязки для выводимого текста.

anchor

[in] Значение из набора 9 предопределенных способов расположения точки привязки выводимого текста. Задаётся комбинацией двух флагов - флага выравнивания текста по горизонтали и флага выравнивания текста по вертикали. Наименования флагов приведены в Примечании.

data[]

[in] Буфер, в который выводится текст. Данный буфер используется для создания графического ресурса.

width

[in] Ширина буфера в точках (пикселях).

height

[in] Высота буфера в точках (пикселях).

color

[in] Цвет текста.

color_format

[in] Формат цвета, задаётся значением из перечисления ENUM_COLOR_FORMAT.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успешного выполнения, иначе false.

Примечание

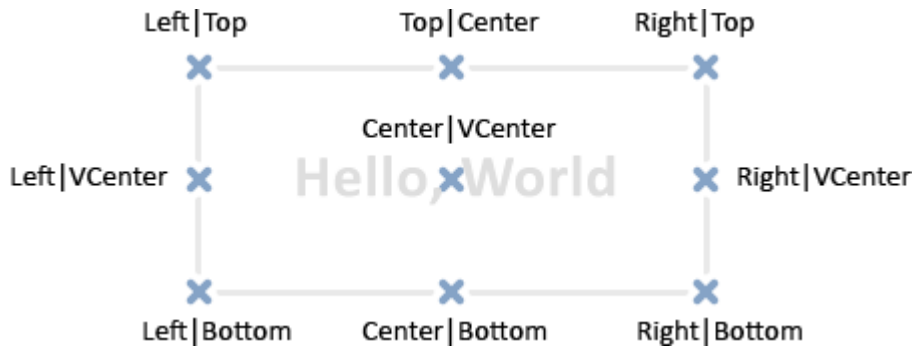
Способ привязки, задаваемый параметром *anchor*, является комбинацией двух флагов выравнивания текста по вертикали и горизонтали. Флаги выравнивания текста по горизонтали:

- TA_LEFT - точка привязки на левой стороне ограничивающего прямоугольника
- TA_CENTER - точка привязки по горизонтали находится в середине ограничивающего прямоугольника
- TA_RIGHT - точка привязки на правой стороне ограничивающего прямоугольника

Флаги выравнивания текста по вертикали:

- TA_TOP - точка привязки на верхней стороне ограничивающего прямоугольника
- TA_VCENTER - точка привязки по вертикали находится в середине ограничивающего прямоугольника
- TA_BOTTOM - точка привязки на нижней стороне ограничивающего прямоугольника

Возможные комбинации флагов и задаваемые ими способы привязки показаны на рисунке.



Пример:

```
//--- ширина и высота канваса (холста, на котором происходит рисование)
#define IMG_WIDTH  200
#define IMG_HEIGHT 200
//--- перед запуском скрипта покажем окно с параметрами
#property script_show_inputs
//--- дадим возможность задавать формат цвета
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- массив (буфер) для отрисовки
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим объект OBJ_BITMAP_LABEL для рисования
   ObjectCreate(0,"CLOCK",OBJ_BITMAP_LABEL,0,0,0);
//--- укажем имя графического ресурса для отрисовки в объекте CLOCK
   ObjectSetString(0,"CLOCK",OBJPROP_BMPFILE,"::IMG");

//--- вспомогательные переменные
```

© 2000-2025, MetaQuotes Ltd.

```
   double a;             // угол стрелки
   uint   nm=2700;       // счетчик минут
   uint   nh=2700*12;    // счетчик часов
   uint   w,h;           // переменные для получения размеров текстовых строк
   int    x,y;           // переменные для расчета текущих координат точки привязки тек

//--- крутим стрелки часов в бесконечном цикле, пока скрипт не остановят
   while(!IsStopped())
     {
      //--- очистка массива буфера рисования часов
      ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
      //--- установка шрифта для отрисовки цифр на циферблате
      TextSetFont("Arial",-200,FW_EXTRABOLD,0);
      //--- рисуем циферблат
      for(int i=1;i<=12;i++)
        {
         //--- получим размеры текущего часа на циферблате
         TextGetSize(string(i),w,h);
         //--- вычислим координаты текущего часа на циферблате
         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);
         //--- вывод этого часа на циферблат в буфер ExtImg[]
         TextOut(string(i),x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,c
        }
      //--- теперь установим шрифт для отрисовки минутной стрелки
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nm%3600));
      //--- получим размеры минутной стрелки
      TextGetSize("----->",w,h);
      //--- вычислим координаты минутной стрелки на циферблате
      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);
      //--- вывод минутной стрелки на циферблат в буфер ExtImg[]
      TextOut("----->",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_f

      //--- теперь установим шрифт для отрисовки часовой стрелки
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nh/12%3600));
      TextGetSize("==>",w,h);
      //--- вычислим координаты часовой стрелки на циферблате
      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);
      //--- вывод часовой стрелки на циферблат в буфер ExtImg[]
      TextOut("==>",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_form

      //--- обновление графического ресурса
      ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
      //--- принудительное обновление графика
```

© 2000-2025, MetaQuotes Ltd.

```
      ChartRedraw();

      //--- увеличим счетчики часа и минут
      nm+=60;
      nh+=60;
      //--- выдержим небольшую паузу между кадрами
      Sleep(10);
     }
//--- удалим объект CLOCK при завершении работы скрипта
   ObjectDelete(0,"CLOCK");
//---
  }
```

Смотри также

Ресурсы, ResourceCreate(), ResourceSave(), TextGetSize(), TextSetFont()

© 2000-2025, MetaQuotes Ltd.

TextGetSize

Возвращает ширину и высоту строки при текущих настройках шрифта.

```
bool  TextGetSize(
   const string       text,          // строка текста
   uint&               width,        // ширина буфера в точках
   uint&               height        // высота буфера в точках
   );
```

Параметры

text

[in] Строка для которой получаем длину и ширину.

width

[out] Входной параметр для получения ширины.

height

[out] Входной параметр для получения высоты.

Возвращаемое значение

Возвращает true в случае успешного выполнения, иначе false. Возможные коды ошибок:

- ERR_INTERNAL_ERROR(4001) - в случае ошибки операционной системы.

Смотри также

Ресурсы, ResourceCreate(), ResourceSave(), TextSetFont(), TextOut()

© 2000-2025, MetaQuotes Ltd.

Функции для работы с техническими индикаторами

Все функции типа iMA, iAC, iMACD, iIchimoku и т.п., создают в глобальном кеше клиентского терминала копию соответствующего технического индикатора. Если копия индикатора с этими параметрами уже существует, то новая копия не создается, а увеличивается счетчик ссылок на данную копию.

Эти функции возвращают хэндл соответствующей копии индикатора. Используя этот хэндл в дальнейшем можно получать данные, рассчитанные соответствующим индикатором. Данные соответствующего буфера (технические индикаторы содержат рассчитанные данные в своих внутренних буферах, которых, в зависимости от индикатора, может быть от 1 до 5) можно копировать в mql5-программу при помощи функции CopyBuffer().

Нельзя обратиться к данным индикатора сразу после его создания, так как на расчет значений индикатора требуется некоторое время, и поэтому создавать хэндлы индикаторов лучше всего в OnInit(). Функция iCustom() создает соответствующий пользовательский индикатор и при успешном создании возвращает его хэндл. Пользовательские индикаторы могут содержать до 512 индикаторных буферов, содержимое которых также можно получать при помощи функции CopyBuffer(), используя полученный хэндл.

Существует универсальный метод создания любого технического индикатора с помощью функции IndicatorCreate(). Эта функция получает в качестве входных параметров:

- имя символа;
- таймфрейм;
- тип создаваемого индикатора;
- количество входных параметров индикатора;
- массив типа MqlParam, содержащий все необходимые входные параметры.

Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание. Многократное обращение к функции индикатора с одними и теми же параметрами в пределах одной mql5-программы не приводит к многократному увеличению счетчика ссылок, счетчик будет увеличен всего один раз на 1. Однако рекомендуется получать хэндлы индикаторов в функции OnInit() или в конструкторе класса, с последующим использованием полученных хэндлов в остальных функциях. Счетчик ссылок уменьшается при деинициализации mql5-программы.

Все индикаторные функции имеют как минимум 2 параметра - символ и период. Значение символа NULL означает текущий инструмент, значение периода 0 означает текущий таймфрейм.

Функция	Возвращает хэндл индикатора:
iAC	Accelerator Oscillator
iAD	Accumulation/Distribution
iADX	Average Directional Index
iADXWilder	Average Directional Index by Welles Wilder
iAlligator	Alligator

© 2000-2025, MetaQuotes Ltd.

Функция	Возвращает хэндл индикатора:
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	пользовательский индикатор
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

© 2000-2025, MetaQuotes Ltd.

Функция	Возвращает хэндл индикатора:
iTriX	Triple Exponential Moving Averages Oscillator
iWPR	Williams' Percent Range
iVIDyA	Variable Index Dynamic Average
iVolumes	Volumes

© 2000-2025, MetaQuotes Ltd.

iAC

Создает в глобальном кеше клиентского терминала индикатор Accelerator Oscillator и возвращает его хэндл. Всего один буфер.

```
int  iAC(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                     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 "Индикатор демонстрирует, как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAC."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- построение 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
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAC,               // использовать iAC
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iAC;          // тип функции
input string                symbol=" ";             // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // таймфрейм
//--- индикаторные буферы
double iACBuffer[];
double iACColors[];
//--- переменная для хранения хэндла индикатора iAC
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Accelerator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iAC)
      handle=iAC(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AC);
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- сообщим о неудаче и выведем номер ошибки
       PrintFormat("Не удалось создать хэндл индикатора iAC для пары %s/%s, код ошибки
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- работа индикатора завершается досрочно
       return(INIT_FAILED);
      }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Accelerator Oscillato
   short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iAC
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iACBuffer больше, чем значений в индикаторе iAC на паре symbol
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы iACBuffer и iACColors значениями из индикатора Accelerator Osc
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем раб
   if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Accelerator Oscillator
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iAC                  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // индикаторный буфер значений Acce
                          double &color_indexes[], // цветовой буфер (для хранения инд
                          int ind_handle,          // хэндл индикатора iAC
                          int amount               // количество копируемых значений
                          )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iACBuffer значениями из индикаторного буфера под индексо
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iAC, код ошибки %d",Get
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- теперь копируем индексы цветов
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- если копирование не удалось,сообщим код ошибки
      PrintFormat("Не удалось скопировать значения цветов из индикатора iAC, код ошиб
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAD

Возвращает хэндл индикатора Accumulation/Distribution. Всего один буфер.

```
int  iAD(
   string               symbol,             // имя символа
   ENUM_TIMEFRAMES      period,             // период
   ENUM_APPLIED_VOLUME  applied_volume      // тип объема для расчета
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

applied_volume

[in] Используемый объем. Может быть любой из ENUM_APPLIED_VOLUME.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                    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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAD."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAD
#property indicator_label1  "iAD"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAD,               // использовать iAD
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iAD;          // тип функции
input ENUM_APPLIED_VOLUME   volumes;                // используемый объем
input string                symbol=" ";             // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // таймфрейм
//--- индикаторный буфер
double          iADBuffer[];
//--- переменная для хранения хэндла индикатора iAD
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Accumulation/Distribution
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iAD)
      handle=iAD(name,period,volumes);
   else
     {
      //--- заполним структуру значениями параметров индикатора
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=volumes;
      handle=IndicatorCreate(name,period,IND_AD,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iAD для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Accumulation/Distribu
   short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iAD
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- если массив iADBuffer больше, чем значений в индикаторе iAD на паре symbol
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iADBuffer значениями из индикатора Accumulation/Distribution
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
   if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Accumulation/Distribution
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iAD                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // индикаторный буфер линии Accumulation/
                         int ind_handle,     // хэндл индикатора iAD
                         int amount          // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iADBuffer значениями из индикаторного буфера под индекс
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iAD, код ошибки %d",Get
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iADX

Возвращает хэндл индикатора Average Directional Movement Index.

```
int  iADX(
   string            symbol,          // имя символа
   ENUM_TIMEFRAMES   period,          // период
   int               adx_period       // период усреднения
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

adx_period

[in] Период для вычисления индекса.

Примечание

Номера буферов: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                    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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iADX."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

© 2000-2025, MetaQuotes Ltd.

```
//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot 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
//--- plot 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
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADX,              // использовать iADX
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iADX;          // тип функции
input int                   adx_period=14;           // период расчета
input string                symbol=" ";              // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;   // таймфрейм
//--- индикаторные буферы
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- переменная для хранения хэндла индикатора iADX
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Average Directional Movement Inde
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iADX)
      handle=iADX(name,period,adx_period);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADX,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iADX для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Average Directional M
   short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_perio
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- количество копируемых значений из индикатора iADX
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив ADXBuffer больше, чем значений в индикаторе iADX на паре symbo
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив значениями из индикатора Average Directional Movement Index
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем раб
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Average Directional Movement Index
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iADX                 |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool FillArraysFromBuffers(double &adx_values[],      // индикаторный буфер линии ADX
                           double &DIplus_values[],   // индикаторный буфер для DI+
                           double &DIminus_values[],  // индикаторный буфер для DI-
                           int ind_handle,            // хэндл индикатора iADXWilder
                           int amount                 // количество копируемых значени
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива ADXBuffer значениями из индикаторного буфера под индексо
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iADX, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива DI_plusBuffer значениями из индикаторного буфера под инд
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iADX, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива DI_plusBuffer значениями из индикаторного буфера под инд
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iADX, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iADXWilder

Возвращает хэндл индикатора Average Directional Movement Index by Welles Wilder.

```
int  iADXWilder(
   string            symbol,          // имя символа
   ENUM_TIMEFRAMES   period,          // период
   int               adx_period       // период усреднения
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

adx_period

[in] Период для вычисления индекса.

Примечание

Номера буферов: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iADXWilder."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

© 2000-2025, MetaQuotes Ltd.

```
//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot 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
//--- plot 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
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADXWilder,         // использовать iADXWilder
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iADXWilder;   // тип функции
input int                   adx_period=14;          // период расчета
input string                symbol=" ";             // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // таймфрейм
//--- индикаторные буферы
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- переменная для хранения хэндла индикатора iADXWilder
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Average Directional Movement Inde
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторномым буферам
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iADXWilder)
      handle=iADXWilder(name,period,adx_period);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iADXWilder для пары %s/%s, код
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Average Directional M
   short_name=StringFormat("iADXWilder(%s/%s period=%d)",name,EnumToString(period),adx
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- количество копируемых значений из индикатора iADXWilder
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив ADXBuffer больше, чем значений в индикаторе iADXWilder на паре
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив значениями из индикатора Average Directional Movement Index by
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Average Directional Movement Index
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iADXWilder           |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool FillArraysFromBuffers(double &adx_values[],      // индикаторный буфер лини ADX
                           double &DIplus_values[],   // индикаторный буфер для DI+
                           double &DIminus_values[],  // индикаторный буфер для DI-
                           int ind_handle,            // хэндл индикатора iADXWilder
                           int amount                 // количество копируемых значени
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива ADXBuffer значениями из индикаторного буфера под индексо
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iADXWilder, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива DI_plusBuffer значениями из индикаторного буфера под инд
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iADXWilder, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива DI_plusBuffer значениями из индикаторного буфера под инд
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iADXWilder, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAlligator

Возвращает хэндл индикатора Alligator.

```
int  iAlligator(
   string               symbol,           // имя символа
   ENUM_TIMEFRAMES      period,           // период
   int                  jaw_period,       // период для расчета челюстей
   int                  jaw_shift,        // смещение челюстей по горизонтали
   int                  teeth_period,     // период для расчета зубов
   int                  teeth_shift,      // смещение зубов по горизонтали
   int                  lips_period,      // период для расчета губ
   int                  lips_shift,       // смещение губ по горизонтали
   ENUM_MA_METHOD       ma_method,        // тип сглаживания
   ENUM_APPLIED_PRICE   applied_price     // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

jaw_period

[in] Период усреднения синей линии (челюсти аллигатора).

jaw_shift

[in] Смещение синей линии относительно графика цены.

teeth_period

[in] Период усреднения красной линии (зубов аллигатора).

teeth_shift

[in] Смещение красной линии относительно графика цены.

lips_period

[in] Период усреднения зеленой линии (губ аллигатора).

lips_shift

[in] Смещение зеленой линии относительно графика цены.

ma_method

[in] Метод усреднения. Может быть любым из значений перечисления ENUM_MA_METHOD.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хэндлом другого индикатора.

© 2000-2025, MetaQuotes Ltd.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

Номера буферов: 0 - GATORJAW_LINE, 1 - GATORTEETH_LINE, 2 - GATORLIPS_LINE.

Пример:

```
//+------------------------------------------------------------------+
//|                                                 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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAlligator."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)"
#property description "Все остальные параметры как в стандартном Аллигаторе."

#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
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAlligator,         // использовать iAlligator
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iAlligator;   // тип функции
input string                symbol=" ";             // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // таймфрейм
input int                   jaw_period=13;          // период для линии Челюстей
input int                   jaw_shift=8;            // смещение линии Челюстей
input int                   teeth_period=8;         // период для линии Зубов
input int                   teeth_shift=5;          // смещение линии Челюстей
input int                   lips_period=5;          // период для линии Губ
input int                   lips_shift=3;           // смещение линии Губ
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // метод усреднения линий Аллигатор
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// тип цены, от которой строится
//--- индикаторные буферы
double          JawsBuffer[];
double          TeethBuffer[];
double          LipsBuffer[];
//--- переменная для хранения хэндла индикатора iAlligator
int     handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Alligator
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторномым буферам
   SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- зададим смещение для каждой линии
   PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
```

© 2000-2025, MetaQuotes Ltd.

```
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   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
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[8];
      //--- периоды и смещения линий Аллигатора
      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;
//--- тип сглаживания
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
//--- тип цены
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
//--- создадим хэндл
      handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iAlligator для пары %s/%s, код
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Alligator
   short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%d)",name,EnumToString(pe
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
```

© 2000-2025, MetaQuotes Ltd.

```
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iAlligator
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив JawsBuffer больше, чем значений в индикаторе iAlligator на пар
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора Alligator
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем раб
   if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,l
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
```

© 2000-2025, MetaQuotes Ltd.

```
                              values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Alligator
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iAlligator           |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &jaws_buffer[],  // индикаторный буфер для линии Чел
                           int j_shift,            // смещение линии Челюстей
                           double &teeth_buffer[], // индикаторный буфер для линии Зуб
                           int t_shift,            // смещение линии Зубов
                           double &lips_buffer[],  // индикаторный буфер для линии Губ
                           int l_shift,            // смещение линии Губ
                           int ind_handle,         // хэндл индикатора iAlligator
                           int amount              // количество копируемых значений
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива JawsBuffer значениями из индикаторного буфера под индекс
   if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива TeethBuffer значениями из индикаторного буфера под индек
   if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива LipsBuffer значениями из индикаторного буфера под индекс
   if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAMA

Возвращает хэндл индикатора Adaptive Moving Average. Всего один буфер.

```
int  iAMA(
   string               symbol,              // имя символа
   ENUM_TIMEFRAMES      period,              // период
   int                  ama_period,          // период AMA
   int                  fast_ma_period,      // период быстрой скользящей
   int                  slow_ma_period,      // период медленной скользящей
   int                  ama_shift,           // смещение индикатора по горизонтали
   ENUM_APPLIED_PRICE   applied_price        // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ama_period

[in] Период вычисления, на котором вычисляется коэффициент эффективности.

fast_ma_period

[in] Быстрый период для вычисления коэффициента сглаживания в моменты быстрого рынка.

slow_ma_period

[in] Медленный период для вычисления коэффициента сглаживания в отсутствии тренда.

ama_shift

[in] Сдвиг индикатора относительно ценового графика.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iAMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)"
#property description "Все остальные параметры как в стандартной AMA."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAMA
#property indicator_label1  "iAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAMA,              // использовать iAMA
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iAMA;          // тип функции
input string                symbol=" ";              // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;   // таймфрейм
input int                   ama_period=15;           // период для расчета
input int                   fast_ma_period=2;        // период быстрой скользящей
input int                   slow_ma_period=30;       // период медленной скользящей
input int                   ama_shift=0;             // смещение по горизонтали
input ENUM_APPLIED_PRICE    applied_price;           // тип цены
//--- индикаторный буфер
double         iAMABuffer[];
//--- переменная для хранения хэндла индикатора iAMA
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Adaptive Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- зададим смещение
   PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iAMA)
      handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,appli
   else
     {
      //--- заполним структуру значениями параметров индикатора
      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;
      //--- тип цены
      pars[4].type=TYPE_INT;
      pars[4].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_AMA,5,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iAMA для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Adaptive Moving Avera
```

© 2000-2025, MetaQuotes Ltd.

```
   short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,d)",name,EnumToString(period),ama_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iAMA
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iAMABuffer больше, чем значений в индикаторе iAMA на паре symb
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора Adaptive Moving Average
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем раб
   if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2025, MetaQuotes Ltd.

```
                              short_name,
                              values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Adaptive Moving Average
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iAMA                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // индикаторный буфер линии AMA
                         int a_shift,           // смещение линии AMA
                         int ind_handle,        // хэндл индикатора iAMA
                         int amount             // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iAMABuffer значениями из индикаторного буфера под индекс
   if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iAMA, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAO

Возвращает хэндл индикатора Awesome Oscillator. Всего один буфер.

```
int  iAO(
   string           symbol,     // имя символа
   ENUM_TIMEFRAMES  period      // период
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                     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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAO."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- построение 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
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAO,               // использовать iAO
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iAO;          // тип функции
input string                symbol=" ";             // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // таймфрейм
//--- индикаторные буферы
double          iAOBuffer[];
double          iAOColors[];
//--- переменная для хранения хэндла индикатора iAO
int     handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Awesome Oscillator
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iAOColors,INDICATOR_COLOR_INDEX);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iAO)
      handle=iAO(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AO);
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iAO для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Awesome Oscillator
   short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iAO
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iAOBuffer больше, чем значений в индикаторе iAO на паре symbol
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы iAOBuffer и iAOColors значениями из индикатора Awesome Oscilla
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем раб
   if(!FillArraysFromBuffer(iAOBuffer,iAOColors,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Awesome Oscillator
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iAO                  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // индикаторный буфер значений Awes
                          double &color_indexes[], // цветовой буфер (для хранения инд
                          int ind_handle,          // хэндл индикатора iAO
                          int amount               // количество копируемых значений
                          )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iAOBuffer значениями из индикаторного буфера под индексо
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iAO, код ошибки %d",Get
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- теперь копируем индексы цветов
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать значения цветов из индикатора iAO, код ошибк
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iATR

Возвращает хэндл индикатора Average True Range. Всего один буфер.

```
int  iATR(
   string           symbol,        // имя символа
   ENUM_TIMEFRAMES  period,        // период
   int              ma_period      // период усреднения
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iATR."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iATR
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iATR,// использовать iATR
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input int                  atr_period=14;          // период для вычисления
input Creation             type=Call_iATR;          // тип функции
input string               symbol=" ";              // символ
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;   // таймфрейм
//--- индикаторный буфер
double         iATRBuffer[];
//--- переменная для хранения хэндла индикатора iAC
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Average True Range
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iATR)
      handle=iATR(name,period,atr_period);
   else
     {
      //--- заполним структуру значениями параметров индикатора
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=atr_period;
      handle=IndicatorCreate(name,period,IND_ATR,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошикби
      PrintFormat("Не удалось создать хэндл индикатора iATR для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Average True Range
   short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iATR
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- если массив iATRBuffer больше, чем значений в индикаторе iATR на паре symb
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iATRBuffer  значениями из индикатора Average True Range
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Average True Range
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iATR                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений ATR
                         int ind_handle,    // хэндл индикатора iATR
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iATRBuffer значениями из индикаторного буфера под индек
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iATR, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBearsPower

Возвращает хэндл индикатора Bears Power. Всего один буфер.

```
int  iBearsPower(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iBearsPower."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iBearsPower
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBearsPower,       // использовать iBearsPower
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iBearsPower;  // тип функции
input int                   ma_period=13;           // период скользящей
input string                symbol=" ";             // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // таймфрейм
//--- индикаторный буфер
double          iBearsPowerBuffer[];
//--- переменная для хранения хэндла индикатора iBearsPower
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Bears Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iBearsPower)
      handle=iBearsPower(name,period,ma_period);
   else
     {
      //--- заполним структуру значениями параметров индикатора
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- период скользящей
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iBearsPower для пары %s/%s, ко
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Bears Power
   short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iBearsPower
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- если массив iBearsPowerBuffer больше, чем значений в индикаторе iBearsPowe
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iBearsPowerBuffer  значениями из индикатора Bears Power
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Bears Power
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iBearsPower           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений Bears Power
                         int ind_handle,    // хэндл индикатора iBearsPower
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iBearsPowerBuffer значениями из индикаторного буфера по
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iBearsPower, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBands

Возвращает хэндл индикатора Bollinger Bands®.

```
int  iBands(
   string               symbol,           // имя символа
   ENUM_TIMEFRAMES      period,           // период
   int                  bands_period,     // период для расчета средней линии
   int                  bands_shift,      // смещение индикатора по горизонтали
   double               deviation,        // кол-во стандартных отклонений
   ENUM_APPLIED_PRICE   applied_price     // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

bands_period

[in] Период усреднения основной линии индикатора.

bands_shift

[in] Сдвиг индикатора относительно ценового графика.

deviation

[in] Отклонение от основной линии.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

Номера буферов: 0 - BASE_LINE, 1 - UPPER_BAND, 2 - LOWER_BAND

Пример:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iBands.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iBands."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- построение Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- построение Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- построение Middle
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBands,             // использовать iBands
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation             type=Call_iBands;          // тип функции
input int                  bands_period=20;           // период скользящей средней
input int                  bands_shift=0;             // сдвиг
input double               deviation=2.0;             // кол-во стандартных отклонений
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // тип цены
input string               symbol=" ";                // символ
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторные буферы
double         UpperBuffer[];
double         LowerBuffer[];
```

© 2000-2025, MetaQuotes Ltd.

```
double         MiddleBuffer[];
//--- переменная для хранения хэндла индикатора iBands
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Bollinger Bands
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- зададим смещение для каждой линии
   PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iBands)
      handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[4];
      //--- период скользящей
      pars[0].type=TYPE_INT;
      pars[0].integer_value=bands_period;
      //--- смещение
      pars[1].type=TYPE_INT;
      pars[1].integer_value=bands_shift;
      //--- количество стандартных отклонений
      pars[2].type=TYPE_DOUBLE;
      pars[2].double_value=deviation;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- тип цены
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iBands для пары %s/%s, код ошиб
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Bollinger Bands
   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);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iBands
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- если размер индикаторных массивов больше, чем значений в индикаторе iBands
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив значениями из индикатора Bollinger Bands
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем раб
   if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,v
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Bollinger Bands
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iBands               |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &base_values[],     // индикаторный буфер средней ли
                           double &upper_values[],    // индикаторный буфер верхней гр
                           double &lower_values[],    // индикаторный буфер нижней гра
                           int shift,                 // смещение
                           int ind_handle,            // хэндл индикатора iBands
                           int amount                 // количество копируемых значени
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива MiddleBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iBands, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- заполняем часть массива UpperBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iBands, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считат
      return(false);
     }

//--- заполняем часть массива LowerBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iBands, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считат
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBullsPower

Возвращает хэндл индикатора Bulls Power. Всего один буфер.

```
int  iBullsPower(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iBullsPower."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iBullsPower
#property indicator_label1  "iBullsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBullsPower,       // использовать iBullsPower
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iBullsPower;  // тип функции
input int                   ma_period=13;           // период скользящей
input string                symbol=" ";             // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // таймфрейм
//--- индикаторный буфер
double          iBullsPowerBuffer[];
//--- переменная для хранения хэндла индикатора iBullsPower
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Bulls Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iBullsPower)
      handle=iBullsPower(name,period,ma_period);
   else
     {
      //--- заполним структуру значениями параметров индикатора
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- период скользящей
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошикби
      PrintFormat("Не удалось создать хэндл индикатора iBullsPower для пары %s/%s, ко
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Bulls Power
   short_name=StringFormat("iBullsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iBullsPower
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- если массив iBullsPowerBuffer больше, чем значений в индикаторе iBullsPowe
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iBullsPowerBuffer  значениями из индикатора Bulls Power
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Bulls Power
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iBullsPower           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений Bulls Power
                         int ind_handle,    // хэндл индикатора iBullsPower
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iBullsPowerBuffer значениями из индикаторного буфера по
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iBullsPower, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

iCCI

Возвращает хэндл индикатора Commodity Channel Index. Всего один буфер.

```
int  iCCI(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   ENUM_APPLIED_PRICE   applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления индикатора.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iCCI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- построение iCCI
#property indicator_label1  "iCCI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- горизонтальные уровни в окне индикатора
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iCCI,              // использовать iCCI
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iCCI;               // тип функции
input int                   ma_period=14;                 // период скользящей средней
input ENUM_APPLIED_PRICE    applied_price=PRICE_TYPICAL;  // тип цены
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторный буфер
double         iCCIBuffer[];
//--- переменная для хранения хэндла индикатора iCCI
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Commodity Channel Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iCCI)
      handle=iCCI(name,period,ma_period,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[2];
      //--- период средней
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- тип цены
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_CCI,2,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iCCI для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор CCI
   short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- количество копируемых значений из индикатора iCCI
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iCCIBuffer больше, чем значений в индикаторе iCCI на паре symb
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iCCIBuffer  значениями из индикатора Commodity Channel Index
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Commodity Channel Index
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iCCI                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений Commodity C
                         int ind_handle,    // хэндл индикатора iCCI
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- заполняем часть массива iCCIBuffer значениями из индикаторного буфера под индек
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iCCI, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iChaikin

Возвращает хэндл индикатора Chaikin Oscillator. Всего один буфер.

```
int  iChaikin(
   string                symbol,              // имя символа
   ENUM_TIMEFRAMES       period,              // период
   int                   fast_ma_period,      // быстрый период
   int                   slow_ma_period,      // медленный период
   ENUM_MA_METHOD        ma_method,           // тип сглаживания
   ENUM_APPLIED_VOLUME   applied_volume       // используемый объем
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

fast_ma_period

[in] Быстрый период усреднения для вычисления индикатора.

slow_ma_period

[in] Медленный период усреднения для вычисления индикатора.

ma_method

[in] Тип усреднения.Может быть любой из констант усреднения ENUM_MA_METHOD.

applied_volume

[in] Используемый объем. Может быть любой из перечисления ENUM_APPLIED_VOLUME.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                 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 "Индикатор демонстрирует как нужно получать данные"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "индикаторных буферов для технического индикатора iChaikin."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iChaikin
#property indicator_label1  "iChaikin"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iChaikin,          // использовать iChaikin
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iChaikin;           // тип функции
input int                   fast_ma_period=3;             // быстрый период
input int                   slow_ma_period=10;            // медленный период
input ENUM_MA_METHOD        ma_method=MODE_EMA;           // тип сглаживания
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // тип объема
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторный буфер
double         iChaikinBuffer[];
//--- переменная для хранения хэндла индикатора iChaikin
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Chaikin Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iChaikin)
      handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_volu
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[4];
      //--- быстрый период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ma_period;
      //--- медленный период
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ma_period;
      //--- тип сглаживания
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- тип объема
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iChaikin для пары %s/%s, код ош
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Chaikin Oscillator
   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);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iChaikin
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iChaikinBuffer больше, чем значений в индикаторе iChaikin на п
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iChaikinBuffer  значениями из индикатора Chaikin Oscillator
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Chaikin Oscillator
```

© 2000-2025, MetaQuotes Ltd.

```
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iChaikin              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений Chaikin Osc
                         int ind_handle,    // хэндл индикатора iChaikin
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iChaikinBuffer значениями из индикаторного буфера под ин
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iChaikin, код ошибки %d
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iCustom

Возвращает хэндл указанного пользовательского индикатора.

```
int  iCustom(
   string            symbol,      // имя символа
   ENUM_TIMEFRAMES   period,      // период
   string            name         // папка/имя_пользовательского индикатора
   ...                            // список входных параметров индикатора
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

name

[in] Имя пользовательского индикатора. Если перед именем указан обратный слеш '\', то EX5-файл индикатора ищется относительно корневой директории MQL5. Таким образом при вызове iCustom(Symbol(), Period(), "\FirstIndicator"...) индикатор будет загружаться как MQL5\FirstIndicator.ex5. Если по этому пути файла не окажется, вернется ошибка 4802 (ERR_INDICATOR_CANNOT_CREATE).

Если же путь не начинается на '\', то поиск и загрузка индикатора производится в следующей последовательности:

- Сначала EX5-файл индикатора ищется в той же папке, где находится EX5-файл вызывающей программы. Например, советник CrossMA.EX5 находится в папке MQL5\Experts\MyExperts и содержит вызов iCustom(Symbol(), Period(), "SecondIndicator"...) - тогда поиск индикатора производится по пути MQL5\Experts\MyExperts\SecondIndicator.ex5.
- Если индикатор не найден в том же каталоге, то поиск производится относительно корневой директории индикаторов MQL5\Indicators. То есть ищется файл MQL5\Indicators\SecondIndicator.ex5. Если индикатор не найден по обоим путям, то функция вернет INVALID_HANDLE и будет взведена ошибка 4802 (ERR_INDICATOR_CANNOT_CREATE).

Если путь к индикатору задан в поддиректории, например, как MyIndicators\ThirdIndicator, то сначала поиск ведется в папке вызывающей программы (советник находится в папке MQL5\Experts\MyExperts) по пути MQL5\Experts\MyExperts\MyIndicators\ThirdIndicator.ex5, а затем в случае неуспеха ищется файл MQL5\Indicators\MyIndicators\ThirdIndicator.ex5. При этом разделителем в пути нужно обязательно указывать двойной обратный слеш '\\', например, iCustom(Symbol(), Period(), "MyIndicators\\ThirdIndicator"...)

...

[in] input-параметры пользовательского индикатора, разделенные запятыми. Тип и порядок следования параметров должен соответствовать. Если параметры не указаны, то будут использованы значения по умолчанию.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

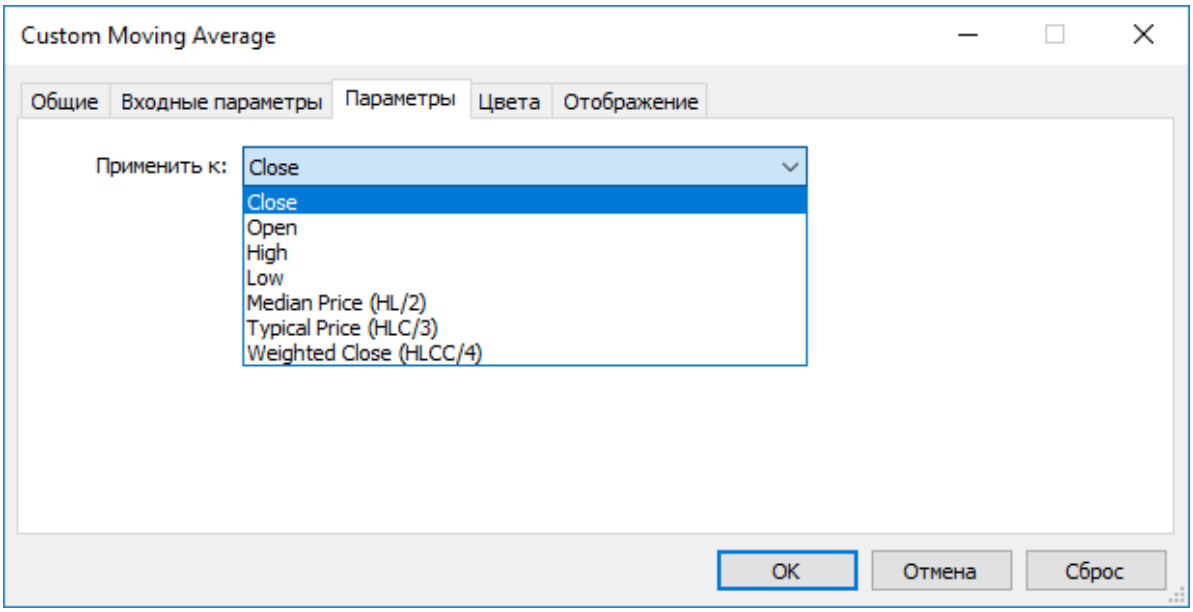
Примечание

Пользовательский индикатор должен быть скомпилирован (файл с расширением EX5) и находиться в директории *MQL5/Indicators* клиентского терминала или вложенной поддиректории.

Необходимые для тестирования индикаторы определяются автоматически из вызова функций iCustom(), если соответствующий параметр задан константной строкой. Для остальных случаев (использование функции IndicatorCreate() или использование неконстантной строки в параметре, задающем имя индикатора) необходимо указать свойство #property tester_indicator:

```
#property tester_indicator "indicator_name.ex5"
```

Если в индикаторе используется первая форма вызова, то при запуске пользовательского индикатора на вкладке "Parameters" можно дополнительно указать на каких данных он будет рассчитываться. Если параметр "Apply to" не выбран явно, то по умолчанию расчет производится по значениям "Close".



При вызове пользовательского индикатора из mql5-программы параметр Applied_Price или хэндл другого индикатора должен передаваться последним после всех предусмотренных пользовательским индикатором входных переменных.

Смотри также

Свойства программ, Доступ к таймсериям и индикаторам, IndicatorCreate(), IndicatorRelease()

Пример:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
```

© 2000-2025, MetaQuotes Ltd.

```
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- indicator buffers
double         Label1Buffer[];
//--- хэндл пользовательского индикатора Custom Moving Average.mq5
int MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
   ResetLastError();
   MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
                     MA_Period,
                     MA_Shift,
                     MA_Method,
                     PRICE_CLOSE // считаем по ценам закрытия
                     );
   Print("MA_handle = ",MA_handle,"  error = ",GetLastError());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- скопируем значения индикатора Custom Moving Average в наш индикаторный буфер
   int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
   Print("copy =",copy,"    rates_total =",rates_total);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если попытка неудачная - сообщим об этом
   if(copy<=0)
      Print("Неудачная попытка получить значения индикатора Custom Moving Average");
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

iDEMA

Возвращает хэндл индикатора Double Exponential Moving Average. Всего один буфер.

```
int  iDEMA(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   int                  ma_shift,          // смещение индикатора по горизонтали
   ENUM_APPLIED_PRICE   applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период(количество баров) для вычисления индикатора.

ma_shift

[in] Сдвиг индикатора относительно ценового графика.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iDEMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iDEMA
#property indicator_label1  "iDEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDEMA,              // использовать iDEMA
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iDEMA;           // тип функции
input int                   ma_period=14;              // период скользящей средней
input int                   ma_shift=0;                // смещение
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // тип цены
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double          iDEMABuffer[];
//--- переменная для хранения хэндла индикатора iDEMA
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Double Exponential Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
   //--- зададим смещение
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iDEMA)
      handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[3];
      //--- период средней
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- смещение
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- тип цены
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iDEMA для пары %s/%s, код ошибк
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Double Exponential Mo
   short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- количество копируемых значений из индикатора iDEMA
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iDEMABuffer больше, чем значений в индикаторе iDEMA на паре sy
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iDEMABuffer  значениями из индикатора Double Exponential Moving
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Double Exponential Moving Average
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iDEMA                 |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений Double Expo
                         int shift,         // смещение
                         int ind_handle,    // хэндл индикатора iDEMA
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iDEMABuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iDEMA, код ошибки %d",G
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iDeMarker

Возвращает хэндл индикатора DeMarker. Всего один буфер.

```
int  iDeMarker(
   string           symbol,        // имя символа
   ENUM_TIMEFRAMES  period,        // период
   int              ma_period      // период усреднения
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iDeMarker."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iDeMarker
#property indicator_label1  "iDeMarker"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- горизонтальные уровни в окне индикатора
#property indicator_level1  0.3
#property indicator_level2  0.7
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDeMarker,          // использовать iDeMarker
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iDeMarker;       // тип функции
input int                   ma_period=14;              // период скользящей
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double          iDeMarkerBuffer[];
//--- переменная для хранения хэндла индикатора iDeMarker
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе DeMarker
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iDeMarker)
      handle=iDeMarker(name,period,ma_period);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[1];
      //--- период средней
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iDeMarker для пары %s/%s, код о
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор DeMarker
   short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma_
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iDeMarker
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iDeMarkerBuffer больше, чем значений в индикаторе iDeMarker на
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iDeMarkerBuffer  значениями из индикатора DeMarker
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе DeMarker
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iDeMarker             |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений DeMarker
                         int ind_handle,    // хэндл индикатора iDeMarker
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iDeMarkerBuffer значениями из индикаторного буфера под и
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iDeMarker, код ошибки %
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
```

© 2000-2025, MetaQuotes Ltd.

```
    return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iEnvelopes

Возвращает хэндл индикатора Envelopes.

```
int  iEnvelopes(
   string               symbol,           // имя символа
   ENUM_TIMEFRAMES      period,           // период
   int                  ma_period,        // период для расчета средней линии
   int                  ma_shift,         // смещение индикатора по горизонтали
   ENUM_MA_METHOD       ma_method,        // тип сглаживания
   ENUM_APPLIED_PRICE   applied_price,    // тип цены или handle
   double               deviation         // отклонение границ от средней линии
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения основной линии индикатора.

ma_shift

[in] Сдвиг индикатора относительно ценового графика.

ma_method

[in] Метод усреднения. Может быть любым из значений значений перечисления ENUM_MA_METHOD.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

deviation

[in] Отклонение от основной линии в процентах.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

Номера буферов: 0 - UPPER_LINE, 1 - LOWER_LINE.

Пример:

```
//+------------------------------------------------------------------+
```

© 2000-2025, 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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iEnvelopes."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- построение Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- построение Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iEnvelopes,        // использовать iEnvelopes
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iEnvelopes;      // тип функции
input int                   ma_period=14;              // период скользящей
input int                   ma_shift=0;                // смещение
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // тип сглаживания
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // тип цены
input double                deviation=0.1;             // отклонение границ от скользящ
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double         UpperBuffer[];
double         LowerBuffer[];
//--- переменная для хранения хэндла индикатора iEnvelopes
```

© 2000-2025, MetaQuotes Ltd.

```
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Envelopes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- зададим смещение для каждой линии
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iEnvelopes)
      handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviati
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[5];
      //--- период средней
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- смещение
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- тип сглаживания
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- тип цены
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      //--- тип цены
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[4].type=TYPE_DOUBLE;
      pars[4].double_value=deviation;
      handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iEnvelopes для пары %s/%s, код
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Envelopes
   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);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iEnvelopes
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- если массив UpperBuffer больше, чем значений в индикаторе iEnvelopes на па
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы UpperBuffer и LowerBuffer значениями из индикатора Envelopes
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Envelopes
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iEnvelopes           |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &upper_values[],     // индикаторный буфер верхней гр
                           double &lower_values[],     // индикаторный буфер нижней гра
                           int shift,                  // смещение
                           int ind_handle,             // хэндл индикатора iEnvelopes
                           int amount                  // количество копируемых значени
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива UpperBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iEnvelopes, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- заполняем часть массива LowerBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iEnvelopes, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iForce

Возвращает хэндл индикатора Force Index. Всего один буфер.

```
int  iForce(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   ENUM_MA_METHOD       ma_method,         // тип сглаживания
   ENUM_APPLIED_VOLUME  applied_volume     // тип объема для расчета
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления индикатора.

ma_method

[in] Метод усреднения. Может быть любым из значений перечисления ENUM_MA_METHOD.

applied_volume

[in] Используемый объем. Может быть любой из значений перечисления ENUM_APPLIED_VOLUME.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iForce."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iForce
#property indicator_label1  "iForce"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iForce,            // использовать iForce
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iForce;             // тип функции
input int                   ma_period=13;                 // период усреднения
input ENUM_MA_METHOD        ma_method=MODE_SMA;           // тип сглаживания
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // тип объема
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторный буфер
double         iForceBuffer[];
//--- переменная для хранения хэндла индикатора iForce
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Force
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
```

© 2000-2025, MetaQuotes Ltd.

```
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iForce)
      handle=iForce(name,period,ma_period,ma_method,applied_volume);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[3];
      //--- период средней
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- тип сглаживания
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_method;
      //--- тип объема
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_volume;
      //--- тип цены
      handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iForce для пары %s/%s, код ошиб
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Force
   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);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- количество копируемых значений из индикатора iForce
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iForceBuffer больше, чем значений в индикаторе iForce на паре
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iForceBuffer  значениями из индикатора Force
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Force
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iForce                |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений Force Index
                         int ind_handle,    // хэндл индикатора iForce
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iForceBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iForce, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iFractals

Возвращает хэндл индикатора Fractals.

```
int  iFractals(
   string           symbol,     // имя символа
   ENUM_TIMEFRAMES  period      // период
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

Номера буферов: 0 - UPPER_LINE, 1 - LOWER_LINE.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iFractals."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- построение FractalUp
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_label1  "FractalUp"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
//--- построение FractalDown
#property indicator_label2  "FractalDown"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFractals,         // использовать iFractals
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iFractals;          // тип функции
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторные буферы
double          FractalUpBuffer[];
double          FractalDownBuffer[];
//--- переменная для хранения хэндла индикатора iFractals
int     handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Fractals
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
//--- зададим коды символом из набора Wingdings для свойств PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,217); // стрелка вверх
   PlotIndexSetInteger(1,PLOT_ARROW,218); // стрелка вниз
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iFractals)
      handle=iFractals(name,period);
   else
      handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iFractals для пары %s/%s, код о
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Fractals
   short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iFractals
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
```

© 2000-2025, MetaQuotes Ltd.

```
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив FractalUpBuffer больше, чем значений в индикаторе iFractals на
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы FractalUpBuffer и FractalDownBuffer значениями из индикатора F
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Fractals
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iFractals            |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &up_arrows[],        // индикаторный буфер стрелок в
                           double &down_arrows[],      // индикаторный буфер стрелок в
                           int ind_handle,             // хэндл индикатора iFractals
                           int amount                  // количество копируемых значен
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива FractalUpBuffer значениями из индикаторного буфера под и
   if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iFractals в массив Frac
                  GetLastError());
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- заполняем часть массива FractalDownBuffer значениями из индикаторного буфера по
   if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iFractals в массив Fra
                  GetLastError());
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iFrAMA

Возвращает хэндл индикатора Fractal Adaptive Moving Average. Всего один буфер.

```
int  iFrAMA(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   int                  ma_shift,          // смещение индикатора по горизонтали
   ENUM_APPLIED_PRICE   applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период(количество баров) для вычисления индикатора.

ma_shift

[in] Сдвиг индикатора относительно ценового графика.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                  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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iFrAMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iFrAMA
#property indicator_label1  "iFrAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFrAMA,            // использовать iFrAMA
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iFrAMA;             // тип функции
input int                   ma_period=14;                 // период усреднения
input int                   ma_shift=0;                   // смещение
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // тип цены
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторный буфер
double         iFrAMABuffer[];
//--- переменная для хранения хэндла индикатора iFrAMA
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Fractal Adaptive Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
//--- зададим смещение
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iFrAMA)
      handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[3];
      //--- период средней
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- смещение
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- тип цены
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      //--- тип цены
      handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iFrAMA для пары %s/%s, код ошиб
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор iFrAMA
   short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2025, 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[])
  {
//--- количество копируемых значений из индикатора iFrAMA
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iFrAMABuffer больше, чем значений в индикаторе iFrAMA на паре
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iFrAMABuffer  значениями из индикатора Fractal Adaptive Moving
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Fractal Adaptive Moving Average
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Заполняем индикаторный буфер из индикатора iFrAMA                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений Fractal Ada
                         int shift,         // смещение
                         int ind_handle,    // хэндл индикатора iFrAMA
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iFrAMABuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iFrAMA, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iGator

Возвращает хэндл индикатора Gator. Осциллятор показывает разницу между синей и красной линией Аллигатора (верхняя гистограмма) и разницу между красной и зеленой линией (нижняя гистограмма).

```
int  iGator(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  jaw_period,        // период для расчета челюстей
   int                  jaw_shift,         // смещение челюстей по горизонтали
   int                  teeth_period,      // период для расчета зубов
   int                  teeth_shift,       // смещение челюстей по зубов
   int                  lips_period,       // период для расчета губ
   int                  lips_shift,        // смещение губ по горизонтали
   ENUM_MA_METHOD       ma_method,         // тип сглаживания
   ENUM_APPLIED_PRICE   applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

jaw_period

[in] Период усреднения синей линии (челюсти аллигатора).

jaw_shift

[in] Смещение синей линии Аллигатора относительно графика цены. Не имеет напрямую отношения к визуальному смещению гистограммы индикатора.

teeth_period

[in] Период усреднения красной линии (зубов аллигатора).

teeth_shift

[in] Смещение красной линии Аллигатора относительно графика цены. Не имеет напрямую отношения к визуальному смещению гистограммы индикатора.

lips_period

[in] Период усреднения зеленой линии (губ аллигатора).

lips_shift

[in] Смещение зеленой линии Аллигатора относительно графика цены. Не имеет напрямую отношения к визуальному смещению гистограммы индикатора.

ma_method

[in] Метод усреднения. Может быть любым из значений ENUM_MA_METHOD.

© 2000-2025, MetaQuotes Ltd.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

Номера буферов: 0 - UPPER_HISTOGRAM, 1- цветовой буфер верхней гистограммы, 2 - LOWER_HISTOGRAM, 3- цветовой буфер нижней гистограммы.

Пример:

```
//+------------------------------------------------------------------+
//|                                                  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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iGator."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Gator Oscillator."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
//--- построение 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
//--- построение 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
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   Call_iGator,            // использовать iGator
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iGator;       // тип функции
input string                symbol=" ";             // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // таймфрейм
input int                   jaw_period=13;          // период для линии Челюстей
input int                   jaw_shift=8;            // смещение линии Челюстей
input int                   teeth_period=8;         // период для линии Зубов
input int                   teeth_shift=5;          // смещение линии Челюстей
input int                   lips_period=5;          // период для линии Губ
input int                   lips_shift=3;           // смещение линии Губ
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // метод усреднения линий Аллигатор
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// тип цены, от которой строится
//--- индикаторные буферы
double          GatorUpBuffer[];
double          GatorUpColors[];
double          GatorDownBuffer[];
double          GatorDownColors[];
//--- переменная для хранения хэндла индикатора iGator
int     handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- значения смещений для верхней и нижней гистограммы
int shift;
//--- будем хранить количество значений в индикаторе Gator Oscillator
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
   SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
   Все указанные в параметрах смещения относятся к индикатору Alligator, по которому с
   Поэтому указанные смещения не производят смещение самого индикатора Gator, а смещае
   по значениям которого строятся значения индикатора Gator Oscillator!
*/
//--- вычислим смещение для верхней и нижней гистограмм, которая является разницей мех
   shift=MathMin(jaw_shift,teeth_shift);
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- несмотря на то, что в индикаторе две гистограммы, используется одинаковое смещен
   PlotIndexSetInteger(1,PLOT_SHIFT,shift);

//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   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
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[8];
      //--- периоды и смещения линий Аллигатора
      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;
      //--- тип сглаживания
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
      //--- тип цены
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
      //--- создадим хэндл
      handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iGator для пары %s/%s, код ошиб
```

© 2000-2025, MetaQuotes Ltd.

```
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Gator Oscillator
   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);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iGator
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив GatorUpBuffer больше, чем значений в индикаторе iGator на паре
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
```

© 2000-2025, MetaQuotes Ltd.

```
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора Gator Oscillator
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем раб
   if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownColo
      shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Gator Oscillator
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iGator               |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &ups_buffer[],         // индикаторный буфер для вер
                           double &up_color_buffer[],    // индикаторный буфер для инд
                           double &downs_buffer[],       // индикаторный буфер для ниж
                           double &downs_color_buffer[], // индикаторный буфер для инд
                           int u_shift,                  // смещение для верхней и ниж
                           int ind_handle,               // хэндл индикатора iGator
                           int amount                    // количество копируемых знач
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива GatorUpBuffer значениями из индикаторного буфера под инд
   if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iGator, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива GatorUpColors значениями из индикаторного буфера под инд
   if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iGator, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- заполняем часть массива GatorDownBuffer значениями из индикаторного буфера под и
   if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iGator, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива GatorDownColors значениями из индикаторного буфера под и
   if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iGator, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iIchimoku

Возвращает хэндл индикатора Ichimoku Kinko Hyo.

```
int  iIchimoku(
   string           symbol,           // имя символа
   ENUM_TIMEFRAMES  period,           // период
   int              tenkan_sen,       // период Tenkan-sen
   int              kijun_sen,        // период Kijun-sen
   int              senkou_span_b     // период Senkou Span B
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

tenkan_sen

[in] Период усреднения Tenkan Sen.

kijun_sen

[in] Период усреднения Kijun Sen.

senkou_span_b

[in] Период усреднения Senkou Span B.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

Номера буферов: 0 - TENKANSEN_LINE, 1 - KIJUNSEN_LINE, 2 - SENKOUSPANA_LINE, 3 - SENKOUSPANB_LINE, 4 - CHIKOUSPAN_LINE.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Индикатор демонстрирует как нужно получать данные"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "индикаторных буферов для технического индикатора iIchimoku."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Ichimoku Kinko Hyo."

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   4
//--- построение 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
//--- построение 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
//--- построение Senkou_Span
#property indicator_label3  "Senkou Span A;Senkou Span B" // в Data Window будет показ
#property indicator_type3   DRAW_FILLING
#property indicator_color3  clrSandyBrown, clrThistle
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//--- построение Chikou_Span
#property indicator_label4  "Chinkou_Span"
#property indicator_type4   DRAW_LINE
#property indicator_color4  clrLime
#property indicator_style4  STYLE_SOLID
#property indicator_width4  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iIchimoku,         // использовать iIchimoku
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation             type=Call_iIchimoku;       // тип функции
input int                  tenkan_sen=9;              // период Tenkan-sen
input int                  kijun_sen=26;              // период Kijun-sen
input int                  senkou_span_b=52;          // период Senkou Span B
input string               symbol=" ";                // символ
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
```

© 2000-2025, MetaQuotes Ltd.

```
double         Tenkan_sen_Buffer[];
double         Kijun_sen_Buffer[];
double         Senkou_Span_A_Buffer[];
double         Senkou_Span_B_Buffer[];
double         Chinkou_Span_Buffer[];
//--- переменная для хранения хэндла индикатора iIchimoku
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Ichimoku Kinko Hyo
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   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);
//--- зададим смещения для канала Senkou Span на kijun_sen баров в будущее
   PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//--- для линии Chikou Span смещение задавать не требуется, так как данные Chikou Span
//--- хранятся в индикаторе iIchimoku уже со смещением
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iIchimoku)
      handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[3];
      //--- периоды и смещения линий Аллигатора
      pars[0].type=TYPE_INT;
      pars[0].integer_value=tenkan_sen;
```

© 2000-2025, 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;
      //--- создадим хэндл
      handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iIchimoku для пары %s/%s, код о
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Ichimoku Kinko Hyo
   short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                           tenkan_sen,kijun_sen,senkou_span_b);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iIchimoku
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
```

© 2000-2025, MetaQuotes Ltd.

```
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив Tenkan_sen_Buffer больше, чем значений в индикаторе iIchimoku
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора Ichimoku Kinko Hyo
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
   if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,S
      kijun_sen,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Ichimoku Kinko Hyo
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iIchimoku            |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[],     // индикаторный буфер лини
                           double &kijun_sen_buffer[],      // индикаторный буфер лини
                           double &senkou_span_A_buffer[],  // индикаторный буфер лини
                           double &senkou_span_B_buffer[],  // индикаторный буфер лини
                           double &chinkou_span_buffer[],   // индикаторный буфер Chi
                           int senkou_span_shift,           // смещение линий Senkou S
                           int ind_handle,                  // хэндл индикатора iIchi
                           int amount                       // количество копируемых з
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива Tenkan_sen_Buffer значениями из индикаторного буфера по
   if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("1.Не удалось скопировать данные из индикатора iIchimoku, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива Kijun_sen_Buffer значениями из индикаторного буфера под
   if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("2.Не удалось скопировать данные из индикатора iIchimoku, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива Chinkou_Span_Buffer значениями из индикаторного буфера п
//--- при senkou_span_shift>0 линия смещена в будущее на senkou_span_shift баров
   if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("3.Не удалось скопировать данные из индикатора iIchimoku, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива Senkou_Span_A_Buffer значениями из индикаторного буфера
//--- при senkou_span_shift>0 линия смещена в будущее на senkou_span_shift баров
   if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("4.Не удалось скопировать данные из индикатора iIchimoku, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива Senkou_Span_B_Buffer значениями из индикаторного буфера
//--- при копировании Chikou Span смещение учитывать не требуется, так как данные Chi
//--- хранятся в индикаторе iIchimoku уже со смещением
   if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("5.Не удалось скопировать данные из индикатора iIchimoku, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBWMFI

Возвращает хэндл индикатора Market Facilitation Index. Всего один буфер.

```
int  iBWMFI(
   string               symbol,              // имя символа
   ENUM_TIMEFRAMES      period,              // период
   ENUM_APPLIED_VOLUME  applied_volume       // тип объема для расчета
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

applied_volume

[in] Используемый объем. Может быть любой из значений перечисления ENUM_APPLIED_VOLUME.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iBWMFI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- построение iBWMFI
#property indicator_label1  "iBWMFI"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_color1  clrLime,clrSaddleBrown,clrBlue,clrPink
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBWMFI,             // использовать iBWMFI
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iBWMFI;          // тип функции
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;// тип объема
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double          iBWMFIBuffer[];
double          iBWMFIColors[];
//--- переменная для хранения хэндла индикатора iBWMFI
int     handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Market Facilitation Index Билла Б
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iBWMFI)
      handle=iBWMFI(name,period,applied_volume);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[1];
      //--- тип объема
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iBWMFI для пары %s/%s, код ошиб
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Market Facilitation I
   short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iBWMFI
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iBWMFIBuffer больше, чем значений в индикаторе iBWMFI на паре
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора Market Facilitation Index Билла Вильям
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем раб
   if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Market Facilitation Index Билла Вильям
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iBWMFI               |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &values[],     // индикаторный буфер значений гистогр
                           double &colors[],     // индикаторный буфер цветов гистограм
                           int ind_handle,       // хэндл индикатора iBWMFI
                           int amount            // количество копируемых значений
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iBWMFIBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iBWMFI, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- заполняем часть массива iBWMFIColors значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iBWMFI, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMomentum

Возвращает хэндл индикатора Momentum. Всего один буфер.

```
int  iMomentum(
   string                symbol,            // имя символа
   ENUM_TIMEFRAMES       period,            // период
   int                   mom_period,        // период усреднения
   ENUM_APPLIED_PRICE    applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

mom_period

[in] Период(количество баров) для вычисления изменения цены.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iMomentum."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)"
#property description "Все остальные параметры как в стандартном Momentum."

#property indicator_separate_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iMomentum
#property indicator_label1  "iMomentum"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMomentum,          // использовать iMomentum
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iMomentum;       // тип функции
input int                   mom_period=14;             // период моментума
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // тип цены
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double          iMomentumBuffer[];
//--- переменная для хранения хэндла индикатора iMomentum
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Momentum
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- создадим хэндл индикатора
   if(type==Call_iMomentum)
      handle=iMomentum(name,period,mom_period,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[2];
      //--- период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=mom_period;
      //--- тип цены
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iMomentum для пары %s/%s, код о
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Momentum
   short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                           mom_period, EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iMomentum
```

© 2000-2025, MetaQuotes Ltd.

```
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iMomentumBuffer больше, чем значений в индикаторе iMomentum на
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iMomentumBuffer  значениями из индикатора Momentum
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Momentum
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iMomentum             |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений Momentum
                         int ind_handle,    // хэндл индикатора iMomentum
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iMomentumBuffer значениями из индикаторного буфера под и
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iMomentum, код ошибки %
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMFI

Расчет Money Flow Index.

```
int  iMFI(
   string                symbol,              // имя символа
   ENUM_TIMEFRAMES       period,              // период
   int                   ma_period,           // период усреднения
   ENUM_APPLIED_VOLUME   applied_volume       // тип объема для расчета
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период(количество баров) для вычисления индикатора.

applied_volume

[in] Используемый объем. Может быть любой из ENUM_APPLIED_VOLUME.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iMFI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)"
#property description "Все остальные параметры как в стандартном Money Flow Index."

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- построение iMFI
#property indicator_label1  "iMFI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- горизонтальные уровни в окне индикатора
#property indicator_level1  20
#property indicator_level2  80
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMFI,              // использовать iMFI
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iMFI;               // тип функции
input int                   ma_period=14;                 // период
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // тип объема
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторный буфер
double         iMFIBuffer[];
//--- переменная для хранения хэндла индикатора iMFI
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Money Flow Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iMFI)
      handle=iMFI(name,period,ma_period,applied_volume);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[2];
      //--- период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- тип объема
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_MFI,2,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iMFI для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Money Flow Index
   short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period, EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- количество копируемых значений из индикатора iMFI
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iMFIBuffer больше, чем значений в индикаторе iMFI на паре symb
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iMFIBuffer  значениями из индикатора Money Flow Index
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Money Flow Index
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iMFI                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // индикаторный буфер значений Money Flow
                         int ind_handle,    // хэндл индикатора iMFI
                         int amount         // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- заполняем часть массива iMFIBuffer значениями из индикаторного буфера под индек
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iMFI, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMA

Возвращает хэндл индикатора скользящего среднего. Всего один буфер.

```
int  iMA(
   string                symbol,            // имя символа
   ENUM_TIMEFRAMES       period,            // период
   int                   ma_period,         // период усреднения
   int                   ma_shift,          // смещение индикатора по горизонтали
   ENUM_MA_METHOD        ma_method,         // тип сглаживания
   ENUM_APPLIED_PRICE    applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления скользящего среднего.

ma_shift

[in] Сдвиг индикатора относительно ценового графика.

ma_method

[in] Метод усреднения. Может быть любым из значений ENUM_MA_METHOD.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                     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"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Moving Average."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iMA
#property indicator_label1  "iMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMA,               // использовать iMA
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iMA;               // тип функции
input int                   ma_period=10;                // период средней
input int                   ma_shift=0;                  // смещение
input ENUM_MA_METHOD        ma_method=MODE_SMA;          // тип сглаживания
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;   // тип цены
input string                symbol=" ";                  // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // таймфрейм
//--- индикаторный буфер
double         iMABuffer[];
//--- переменная для хранения хэндла индикатора iMA
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- зададим смещение
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iMA)
      handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[4];
      //--- период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- смещение
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- тип сглаживания
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- тип цены
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MA,4,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iMA для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Moving Average
   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);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iMA
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iMABuffer больше, чем значений в индикаторе iMA на паре symbol
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив iMABuffer  значениями из индикатора Moving Average
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Moving Average
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iMA                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // индикаторный буфер значений Moving Ave
                         int shift,          // смещение
                         int ind_handle,     // хэндл индикатора iMA
                         int amount          // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iMABuffer значениями из индикаторного буфера под индексо
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iMA, код ошибки %d",Get
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iOsMA

Возвращает хэндл индикатора Moving Average of Oscillator. Осциллятор OsMA показывает разницу между значениями MACD и его сигнальной линии. Всего один буфер.

```
int  iOsMA(
   string               symbol,              // имя символа
   ENUM_TIMEFRAMES      period,              // период
   int                  fast_ema_period,     // период быстрой средней
   int                  slow_ema_period,     // период медленной средней
   int                  signal_period,       // период усреднения разности
   ENUM_APPLIED_PRICE   applied_price        // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

fast_ema_period

[in] Период усреднения для вычисления быстрой скользящей средней.

slow_ema_period

[in] Период усреднения для вычисления медленной скользящей средней.

signal_period

[in] Период усреднения для вычисления сигнальной линии.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

В некоторых системах этот осциллятор называется гистограммой MACD.

Пример:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iOsMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iOsMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Moving Average of Os

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iOsMA
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOsMA,             // использовать iOsMA
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iOsMA;           // тип функции
input int                   fast_ema_period=12;        // период быстрой средней
input int                   slow_ema_period=26;        // период медленной средней
input int                   signal_period=9;           // период усреднения разности
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // тип цены
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double         iOsMABuffer[];
//--- переменная для хранения хэндла индикатора iAMA
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Moving Average of Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iOsMA)
      handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[4];
      //--- быстрый период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- медленный период
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- период усреднения разницы между быстрой и медленной средними
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- тип цены
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iOsMA для пары %s/%s, код ошибк
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Moving Average of Osc
   short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
```

© 2000-2025, MetaQuotes Ltd.

```
                              fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iOsMA
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iOsMABuffer больше, чем значений в индикаторе iOsMA на паре sy
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора iOsMA
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2025, MetaQuotes Ltd.

```
                              short_name,
                              values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе  Moving Average of Oscillator
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iOsMA                 |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // индикаторный буфер значений OsMA
                         int ind_handle,        // хэндл индикатора iOsMA
                         int amount             // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iOsMABuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iOsMA, код ошибки %d",G
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMACD

Возвращает хэндл индикатора Moving Averages Convergence/Divergence. В тех системах, где OsMA называют гистограммой MACD, данный индикатор изображается в виде двух линий. В клиентском терминале схождение/расхождение скользящих средних рисуется в виде гистограммы.

```
int  iMACD(
   string               symbol,              // имя символа
   ENUM_TIMEFRAMES      period,              // период
   int                  fast_ema_period,     // период быстрой средней
   int                  slow_ema_period,     // период медленной средней
   int                  signal_period,       // период усреднения разности
   ENUM_APPLIED_PRICE   applied_price        // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

fast_ema_period

[in] Период усреднения для вычисления быстрой скользящей средней.

slow_ema_period

[in] Период усреднения для вычисления медленной скользящей средней.

signal_period

[in] Период усреднения для вычисления сигнальной линии.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

Номера буферов: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMACD.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
```

© 2000-2025, 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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iMACD."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном MACD."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- построение MACD
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- построение Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+---------------------------------------------------------------------+
//| Перечисление способов создания хэндла                               |
//+---------------------------------------------------------------------+
enum Creation
  {
   Call_iMACD,              // использовать iMACD
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iMACD;           // тип функции
input int                   fast_ema_period=12;        // период быстрой средней
input int                   slow_ema_period=26;        // период медленной средней
input int                   signal_period=9;           // период усреднения разности
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // тип цены
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторные буферы
double         MACDBuffer[];
double         SignalBuffer[];
//--- переменная для хранения хэндла индикатора iMACD
int    handle;
//--- переменная для хранения
```

© 2000-2025, MetaQuotes Ltd.

```
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Moving Averages Convergence/Divei
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iMACD)
      handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[4];
      //--- быстрый период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- медленный период
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- период усреднения разницы между быстрой и медленной средними
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- тип цены
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MACD,4,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Не удалось создать хэндл индикатора iMACD для пары %s/%s, код ошибк
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Moving Averages Conve
   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);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iMACD
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив MACDBuffer больше, чем значений в индикаторе iMACD на паре sym
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора iMACD
//--- если FillArraysFromBuffers вернула false, значит данные не готовы - завершаем ра
   if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0)
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе  Moving Averages Convergence/Divergenc
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iMACD                |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &macd_buffer[],    // индикаторный буфер значений MA
                           double &signal_buffer[],  // индикаторный буфер сигнальной
                           int ind_handle,           // хэндл индикатора iMACD
                           int amount                // количество копируемых значений
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iMACDBuffer значениями из индикаторного буфера под индек
   if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iMACD, код ошибки %d",G
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }

//--- заполняем часть массива SignalBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iMACD, код ошибки %d",G
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iOBV

Возвращает хэндл индикатора On Balance Volume. Всего один буфер.

```
int  iOBV(
   string                symbol,              // имя символа
   ENUM_TIMEFRAMES       period,              // период
   ENUM_APPLIED_VOLUME   applied_volume       // тип объема для расчета
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

applied_volume

[in] Используемый объем. Может быть любой из значений перечисления ENUM_APPLIED_VOLUME.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iOBV."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---  iOBV
#property indicator_label1  "iOBV"
#property indicator_type1   DRAW_LINE
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOBV ,              // использовать iOBV
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iOBV;                // тип функции
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;    // тип объема
input string                symbol=" ";                    // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;         // таймфрейм
//--- индикаторные буферы
double          iOBVBuffer[];
//--- переменная для хранения хэндла индикатора iOBV
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе On Balance Volume
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iOBV)
      handle=iOBV(name,period,applied_volume);
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[1];
      //--- тип объема
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_OBV,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iOBV для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор On Balance Volume
   short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iOBV
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
```

© 2000-2025, MetaQuotes Ltd.

```
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iOBVBuffer больше, чем значений в индикаторе iOBV на паре symb
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора iOBV
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе  On Balance Volume
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iOBV                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &obv_buffer[],  // индикаторный буфер значений OBV
                         int ind_handle,        // хэндл индикатора iOBV
                         int amount             // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iOBVBuffer значениями из индикаторного буфера под индек
   if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iOBV, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iSAR

Возвращает хэндл индикатора Parabolic Stop and Reverse system. Всего один буфер.

```
int  iSAR(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period,      // период
   double           step,        // шаг изменения цены - коэффициент ускорения
   double           maximum      // максимальный шаг
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

step

[in] Шаг изменения цены, обычно 0.02.

maximum

[in] Максимальный шаг, обычно 0.2.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                    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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iSAR."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Parabolic Stop and Re

#property indicator_chart_window
#property indicator_buffers 1
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- построение iSAR
#property indicator_label1  "iSAR"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iSAR,               // использовать iSAR
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iSAR;               // тип функции
input double                step=0.02;                    // шаг - коэффициент ускорени
input double                maximum=0.2;                  // максимальное значение шага
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторные буферы
double         iSARBuffer[];
//--- переменная для хранения хэндла индикатора iSAR
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Parabolic SAR
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
//--- установим свойству PLOT_ARROW код символа из набора Wingdings для отображения на
   PlotIndexSetInteger(0,PLOT_ARROW,159);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
```

© 2000-2025, MetaQuotes Ltd.

```
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iSAR)
      handle=iSAR(name,period,step,maximum);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[2];
      //--- значение шага
      pars[0].type=TYPE_DOUBLE;
      pars[0].double_value=step;
      //--- предельное значение шага, которое может использоваться при расчетах
      pars[1].type=TYPE_DOUBLE;
      pars[1].double_value=maximum;
      handle=IndicatorCreate(name,period,IND_SAR,2,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iSAR для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Parabolic SAR
   short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                           step,maximum);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- количество копируемых значений из индикатора iSAR
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iSARBuffer больше, чем значений в индикаторе iSAR на паре symb
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора iSAR
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Parabolic SAR
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iSAR                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &sar_buffer[],  // индикаторный буфер значений Parabol
                         int ind_handle,        // хэндл индикатора iSAR
                         int amount             // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- заполняем часть массива iSARBuffer значениями из индикаторного буфера под индекс
   if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iSAR, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iRSI

Возвращает хэндл индикатора Relative Strength Index. Всего один буфер.

```
int  iRSI(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   ENUM_APPLIED_PRICE   applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления индекса.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iRSI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Relative Strength Inc

#property indicator_separate_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iRSI
#property indicator_label1  "iRSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- пределы для отображения значений в окне индикатора
#property indicator_maximum 100
#property indicator_minimum 0
//--- горизонтальные уровни в окне индикатора
#property indicator_level1  70.0
#property indicator_level2  30.0
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRSI,              // использовать iRSI
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iRSI;               // тип функции
input int                   ma_period=14;                 // период усреднения
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // тип цены
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторный буфер
double          iRSIBuffer[];
//--- переменная для хранения хэндла индикатора iRSI
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Relative Strength Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iRSI)
      handle=iRSI(name,period,ma_period,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[2];
      //--- период средней
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- предельное значение шага, которое может использоваться при расчетах
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_RSI,2,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iRSI для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Relative Strength Inc
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                           ma_period,applied_price);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- количество копируемых значений из индикатора iRSI
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iRSIBuffer больше, чем значений в индикаторе iRSI на паре symb
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив значениями из индикатора iRSI
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Relative Strength Index
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iRSI                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rsi_buffer[],  // индикаторный буфер значений Relativ
                         int ind_handle,        // хэндл индикатора iRSI
```

© 2000-2025, MetaQuotes Ltd.

```
                           int amount              // количество копируемых значений
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iRSIBuffer значениями из индикаторного буфера под индек
   if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iRSI, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iRVI

Возвращает хэндл индикатора Relative Vigor Index.

```
int  iRVI(
   string           symbol,         // имя символа
   ENUM_TIMEFRAMES  period,         // период
   int              ma_period       // период усреднения
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления индекса.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

Номера буферов: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Пример:

```
//+------------------------------------------------------------------+
//|                                                    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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iRVI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Relative Vigor Index.

#property indicator_separate_window
#property indicator_buffers 2
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   2
//--- построение RVI
#property indicator_label1  "RVI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- построение Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRVI,              // использовать iRVI
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iRVI;               // тип функции
input int                   ma_period=10;                 // период для расчетов
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторные буферы
double          RVIBuffer[];
double          SignalBuffer[];
//--- переменная для хранения хэндла индикатора iRVI
int     handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Relative Vigor Index
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iRVI)
      handle=iRVI(name,period,ma_period);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[1];
      //--- период для расчетов
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_RVI,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iRVI для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Relative Vigor Index
   short_name=StringFormat("iRVI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- количество копируемых значений из индикатора iRVI
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив RVIBuffer больше, чем значений в индикаторе iRVI на паре symbo
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора iRVI
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Relative Vigor Index
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iRVI                 |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rvi_buffer[],     // индикаторный буфер значений Rela
                         double &signal_buffer[],  // индикаторный буфер сигнальной ли
                         int ind_handle,           // хэндл индикатора iRVI
                         int amount                // количество копируемых значений
                         )
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iRVIBuffer значениями из индикаторного буфера под индек
   if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iRVI, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- заполняем часть массива SignalBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iRVI, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iStdDev

Возвращает хэндл индикатора Standard Deviation. Всего один буфер.

```
int  iStdDev(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   int                  ma_shift,          // смещение индикатора по горизонтали
   ENUM_MA_METHOD       ma_method,         // тип сглаживания
   ENUM_APPLIED_PRICE   applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период усреднения для вычисления индикатора.

ma_shift

[in] Сдвиг индикатора относительно ценового графика.

ma_method

[in] Метод усреднения. Может быть любым из значений ENUM_MA_METHOD.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iStdDev."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)"
#property description "Все остальные параметры как в стандартном Standard Deviation."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iStdDev
#property indicator_label1  "iStdDev"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStdDev,           // использовать iStdDev
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iStdDev;         // тип функции
input int                   ma_period=20;              // период усреднения
input int                   ma_shift=0;                // смещение
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // тип сглаживания
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // тип цены
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double         iStdDevBuffer[];
//--- переменная для хранения хэндла индикатора iStdDev
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Standard Deviation
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- зададим смещение
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iStdDev)
      handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[4];
      //--- период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- смещение
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- тип сглаживания
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- тип цены
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iStdDev для пары %s/%s, код ош
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Standard Deviation
   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);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iStdDev
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iStdDevBuffer больше, чем значений в индикаторе iStdDev на пар
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив значениями из индикатора Standard Deviation
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Standard Deviation
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iStdDev               |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &std_buffer[],  // индикаторный буфер линии Standard D
                         int std_shift,         // смещение линии Standard Deviation
                         int ind_handle,        // хэндл индикатора iStdDev
                         int amount             // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iStdDevBuffer значениями из индикаторного буфера под инд
   if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iStdDev, код ошибки %d"
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iStochastic

Возвращает хэндл индикатора Stochastic Oscillator.

```
int  iStochastic(
   string            symbol,          // имя символа
   ENUM_TIMEFRAMES   period,          // период
   int               Kperiod,         // K-период (количество баров для расчетов)
   int               Dperiod,         // D-период (период первичного сглаживания)
   int               slowing,         // окончательное сглаживание
   ENUM_MA_METHOD    ma_method,       // тип сглаживания
   ENUM_STO_PRICE    price_field      // способ расчета стохастика
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

Kperiod

[in] K-период(количество баров) для вычисления линии %K.

Dperiod

[in] Период усреднения для вычисления линии %D.

slowing

[in] Значение замедления.

ma_method

[in] Метод усреднения. Может быть любым из значений ENUM_MA_METHOD.

price_field

[in] Параметр выбора цен для расчета. Может быть одной из величин ENUM_STO_PRICE.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Примечание

Номера буферов: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Пример:

```
//+------------------------------------------------------------------+
//|                                              Demo_iStochastic.mq5 |
```

© 2000-2025, 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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iStochastic."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Stochastic Oscillato

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- построение Stochastic
#property indicator_label1  "Stochastic"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- построение Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- зададим граничные значения индикатора
#property indicator_minimum 0
#property indicator_maximum 100
//--- горизонтальные уровни в окне индикатора
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStochastic,       // использовать iStochastic
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation             type=Call_iStochastic;     // тип функции
input int                  Kperiod=5;                 // K-период (количество баров дл
input int                  Dperiod=3;                 // D-период (период первичного с
input int                  slowing=3;                 // период для окончательного сгл
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // тип сглаживания
input ENUM_STO_PRICE       price_field=STO_LOWHIGH;   // способ расчета стохастика
```

© 2000-2025, MetaQuotes Ltd.

```
input string               symbol=" ";                // символ
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторные буферы
double         StochasticBuffer[];
double         SignalBuffer[];
//--- переменная для хранения хэндла индикатора iStochastic
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Stochastic Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массивов к индикаторным буферам
   SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iStochastic)
      handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[5];
      //--- период K для расчетов
      pars[0].type=TYPE_INT;
      pars[0].integer_value=Kperiod;
      //--- период D для первичного сглаживания
      pars[1].type=TYPE_INT;
      pars[1].integer_value=Dperiod;
      //--- период K для окончательного сглаживания
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slowing;
      //--- тип сглаживания
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ma_method;
      //--- способ расчета стохастика
      pars[4].type=TYPE_INT;
      pars[4].integer_value=price_field;
      handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iStochastic для пары %s/%s, ко
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Stochastic Oscillato
   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);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iStochastic
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
```

© 2000-2025, MetaQuotes Ltd.

```
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив StochasticBuffer больше, чем значений в индикаторе iStochastic
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора iStochastic
//--- если FillArraysFromBuffers вернула false, значит данные не готовы - завершаем ра
   if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) ret
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Stochastic Oscillator
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iStochastic          |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &main_buffer[],    // индикаторный буфер значений St
                           double &signal_buffer[],  // индикаторный буфер сигнальной
                           int ind_handle,           // хэндл индикатора iStochastic
                           int amount                // количество копируемых значений
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива StochasticBuffer значениями из индикаторного буфера под
   if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iStochastic, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- заполняем часть массива SignalBuffer значениями из индикаторного буфера под инде
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iStochastic, код ошибки
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iTEMA

Возвращает хэндл индикатора Triple Exponential Moving Average. Всего один буфер.

```
int  iTEMA(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   int                  ma_shift,          // смещение индикатора по горизонтали
   ENUM_APPLIED_PRICE   applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период(количество баров) для вычисления индикатора.

ma_shift

[in] Сдвиг индикатора относительно ценового графика.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iTEMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Triple Exponential Mo

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iTEMA
#property indicator_label1  "iTEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTEMA,              // использовать iTEMA
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iTEMA;           // тип функции
input int                   ma_period=14;              // период усреднения
input int                   ma_shift=0;                // смещение
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // тип цены
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double         iTEMABuffer[];
//--- переменная для хранения хэндла индикатора iTEMA
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Triple Exponential Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
//--- зададим смещение
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iTEMA)
      handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[3];
      //--- период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- смещение
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- тип цены
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iTEMA для пары %s/%s, код ошибк
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Triple Exponential Mo
   short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2025, 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[])
  {
//--- количество копируемых значений из индикатора iTEMA
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iTEMABuffer больше, чем значений в индикаторе iTEMA на паре sy
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив значениями из индикатора Triple Exponential Moving Average
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Triple Exponential Moving Average
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Заполняем индикаторный буфер из индикатора iTEMA                 |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &tema_buffer[], // индикаторный буфер значений Triple
                         int t_shift,           // смещение линии
                         int ind_handle,        // хэндл индикатора iTEMA
                         int amount             // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iTEMABuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iTEMA, код ошибки %d",G
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iTriX

Возвращает хэндл индикатора Triple Exponential Moving Averages Oscillator. Всего один буфер.

```
int  iTriX(
   string               symbol,            // имя символа
   ENUM_TIMEFRAMES      period,            // период
   int                  ma_period,         // период усреднения
   ENUM_APPLIED_PRICE   applied_price      // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

ma_period

[in] Период(количество баров) для вычисления индикатора.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iTriX."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- построение iTriX
#property indicator_label1  "iTriX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTriX,              // использовать iTriX
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iTriX;           // тип функции
input int                   ma_period=14;              // период
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // тип цены
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double         iTriXBuffer[];
//--- переменная для хранения хэндла индикатора iTriX
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Triple Exponential Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- создадим хэндл индикатора
   if(type==Call_iTriX)
      handle=iTriX(name,period,ma_period,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[2];
      //--- период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- тип цены
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iTriX для пары %s/%s, код ошибк
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Triple Exponential Mo
   short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iTriX
   int values_to_copy;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iTriXBuffer больше, чем значений в индикаторе iTriX на паре sy
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив значениями из индикатора Triple Exponential Moving Averages Osc
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Triple Exponential Moving Averages Osc
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iTriX                 |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &trix_buffer[], // индикаторный буфер значений Triple
                         int ind_handle,        // хэндл индикатора iTriX
                         int amount             // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iTriXBuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- если копирование не удалось, сообщим код ошибки
       PrintFormat("Не удалось скопировать данные из индикатора iTriX, код ошибки %d",G
       //--- завершим с нулевым результатом - это означает, что индикатор будет считать
       return(false);
      }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iWPR

Возвращает хэндл индикатора Larry Williams' Percent Range. Всего один буфер.

```
int  iWPR(
   string           symbol,          // имя символа
   ENUM_TIMEFRAMES  period,          // период
   int              calc_period      // период усреднения
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

calc_period

[in] Период(количество баров) для вычисления индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                   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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iWPR."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iWPR
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrCyan
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- зададим граничные значения индикатора
#property indicator_minimum -100
#property indicator_maximum 0
//--- горизонтальные уровни в окне индикатора
#property indicator_level1  -20.0
#property indicator_level2  -80.0
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iWPR,              // использовать iWPR
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iWPR;             // тип функции
input int                   calc_period=14;             // период
input string                symbol=" ";                 // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // таймфрейм
//--- индикаторный буфер
double         iWPRBuffer[];
//--- переменная для хранения хэндла индикатора iWPR
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Larry Williams' Percent Range
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- создадим хэндл индикатора
   if(type==Call_iWPR)
      handle=iWPR(name,period,calc_period);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[1];
      //--- период
      pars[0].type=TYPE_INT;
      pars[0].integer_value=calc_period;
      handle=IndicatorCreate(name,period,IND_WPR,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iWPR для пары %s/%s, код ошибки
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Williams' Percent Ran
   short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iWPR
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iWPRBuffer больше, чем значений в индикаторе iWPR на паре symb
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив значениями из индикатора Williams' Percent Range
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Williams' Percent Range
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iWPR                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &wpr_buffer[], // индикаторный буфер значений Williams
                         int ind_handle,       // хэндл индикатора iWPR
                         int amount            // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iWPRBuffer значениями из индикаторного буфера под индек
   if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iWPR, код ошибки %d",Ge
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iVIDyA

Возвращает хэндл индикатора Variable Index Dynamic Average. Всего один буфер.

```
int  iVIDyA(
   string               symbol,           // имя символа
   ENUM_TIMEFRAMES      period,           // период
   int                  cmo_period,       // период Chande Momentum
   int                  ema_period,       // период фактора сглаживания
   int                  ma_shift,         // смещение индикатора по горизонтали
   ENUM_APPLIED_PRICE   applied_price     // тип цены или handle
   );
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

cmo_period

[in] Период(количество баров) для вычисления Chande Momentum Oscillator.

ema_period

[in] Период(количество баров) EMA для вычисления фактора сглаживания.

ma_shift

[in] Сдвиг индикатора относительно ценового графика.

applied_price

[in] Используемая цена. Может быть любой из ценовых констант ENUM_APPLIED_PRICE или хендлом другого индикатора.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                 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"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iVIDyA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)
#property description "Все остальные параметры как в стандартном Variable Index Dynami

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iVIDyA
#property indicator_label1  "iVIDyA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVIDyA,            // использовать iVIDyA
   Call_IndicatorCreate    // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iVIDyA;          // тип функции
input int                   cmo_period=15;             // период Chande Momentum
input int                   ema_period=12;             // период фактора сглаживания
input int                   ma_shift=0;                // смещение
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // тип цены
input string                symbol=" ";                // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double         iVIDyABuffer[];
//--- переменная для хранения хэндла индикатора iVIDyA
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Variable Index Dynamic Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторному буферу
   SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- зададим смещение
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iVIDyA)
      handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[4];
      //--- период Chande Momentum
      pars[0].type=TYPE_INT;
      pars[0].integer_value=cmo_period;
      //--- период фактора сглаживания
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ema_period;
      //--- смещение
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_shift;
      //--- тип цены
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iVIDyA для пары %s/%s, код ошиб
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Variable Index Dynami
   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);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iVIDyA
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iVIDyABuffer больше, чем значений в индикаторе iVIDyA на паре
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массив значениями из индикатора Variable Index Dynamic Average
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем рабо
   if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Variable Index Dynamic Average
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторный буфер из индикатора iVIDyA                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &vidya_buffer[],// индикаторный буфер значений Variabl
                         int v_shift,           // смещение линии
                         int ind_handle,        // хэндл индикатора iVIDyA
                         int amount             // количество копируемых значений
                         )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iVIDyABuffer значениями из индикаторного буфера под инде
   if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iVIDyA, код ошибки %d",
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iVolumes

Возвращает хэндл индикатора, отображающего объемы . Всего один буфер.

```
int  iVolumes(
   string               symbol,              // имя символа
   ENUM_TIMEFRAMES      period,              // период
   ENUM_APPLIED_VOLUME  applied_volume       // тип объема
   )
```

Параметры

symbol

[in] Символьное имя инструмента, на данных которого будет вычисляться индикатор. NULL означает текущий символ.

period

[in] Значение периода может быть одним из значений перечисления ENUM_TIMEFRAMES, 0 означает текущий таймфрейм.

applied_volume

[in] Используемый объем. Может быть любой из значений перечисления ENUM_APPLIED_VOLUME.

Возвращаемое значение

Возвращает хэндл указанного технического индикатора, в случае неудачи возвращает INVALID_HANDLE. Для освобождения памяти компьютера от неиспользуемого больше индикатора служит функция IndicatorRelease(), которой передается хэндл этого индикатора.

Пример:

```
//+------------------------------------------------------------------+
//|                                                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 "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iVolumes."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- построение iVolumes
#property indicator_label1  "iVolumes"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Перечисление способов создания хэндла                            |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVolumes,           // использовать iVolumes
   Call_IndicatorCreate     // использовать IndicatorCreate
  };
//--- входные параметры
input Creation              type=Call_iVolumes;           // тип функции
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // тип объема
input string                symbol=" ";                   // символ
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // таймфрейм
//--- индикаторные буферы
double          iVolumesBuffer[];
double          iVolumesColors[];
//--- переменная для хранения хэндла индикатора iVolumes
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Volumes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- привязка массива к индикаторным буферам
   SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
//--- определимся с символом, на котором строится индикатор
   name=symbol;
//--- удалим пробелы слева и справа
   StringTrimRight(name);
   StringTrimLeft(name);
//--- если после этого длина строки name нулевая
   if(StringLen(name)==0)
     {
      //--- возьмем символ с графика, на котором запущен индикатор
      name=_Symbol;
     }
//--- создадим хэндл индикатора
   if(type==Call_iVolumes)
      handle=iVolumes(name,period,applied_volume);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- заполним структуру значениями параметров индикатора
      MqlParam pars[1];
      //--- тип цены
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
     }
//--- если не удалось создать хэндл
   if(handle==INVALID_HANDLE)
     {
      //--- сообщим о неудаче и выведем номер ошибки
      PrintFormat("Не удалось создать хэндл индикатора iVolumes для пары %s/%s, код ош
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- работа индикатора завершается досрочно
      return(INIT_FAILED);
     }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Volumes
   short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- количество копируемых значений из индикатора iVolumes
   int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
      return(0);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- если это первый запуск вычислений нашего индикатора или изменилось количество зн
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- если массив iVolumesBuffer больше, чем значений в индикаторе iVolumes на п
      //--- в противном случае копировать будем меньше, чем размер индикаторных буферо
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
      //--- для расчета добавилось не более одного бара
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- заполняем массивы значениями из индикатора iVolumes
//--- если FillArraysFromBuffers вернула false, значит данные не готовы - завершаем ра
   if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) ret
//--- сформируем сообщение
   string comm=StringFormat("%s ==>  Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
   Comment(comm);
//--- запомним количество значений в индикаторе Volumes
   bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Заполняем индикаторные буферы из индикатора iVolumes             |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &volume_buffer[],   // индикаторный буфер значений V
                           double &color_buffer[],    // индикаторный буфер цветов
                           int ind_handle,            // хэндл индикатора iVolumes
                           int amount                 // количество копируемых значени
                           )
  {
//--- сбросим код ошибки
   ResetLastError();
//--- заполняем часть массива iVolumesBuffer значениями из индикаторного буфера под ин
   if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iVolumes, код ошибки %d
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- заполняем часть массива iVolumesColors значениями из индикаторного буфера под ин
   if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
     {
      //--- если копирование не удалось, сообщим код ошибки
      PrintFormat("Не удалось скопировать данные из индикатора iVolumes, код ошибки %d
      //--- завершим с нулевым результатом - это означает, что индикатор будет считать
      return(false);
     }
//--- все получилось
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- почистим график при удалении индикатора
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

Работа с результатами оптимизации

Функции для организации собственной обработки результатов оптимизации в тестере стратегий. Могут вызываться при оптимизации в агентах тестирования, а также локально в экспертах и скриптах.

При запуске эксперта в тестере стратегий можно создать собственный массив данных на основе простых типов или простых структур (не содержат строки, объекты класса или объекты динамических массивов). Этот набор данных можно сохранить с помощью функции FrameAdd() в специальной структуре, называемой фрейм (кадр). Каждый агент при оптимизации эксперта может посылать в терминал серию фреймов. Все полученные фреймы в порядке поступления от агентов записываются в *.MQD-файл по имени эксперта в папку каталог_терминала\MQL5\Files\Tester. Поступление фрейма в клиентский терминал от агента тестирования генерирует событие TesterPass.

Фреймы можно сохранять как в память компьютера, так и в файл с указанным именем. Нет никаких ограничений на количество фреймов со стороны языка MQL5.

Ограничения на память и дисковое пространство в MQL5 Cloud Network

При запуске оптимизации в MQL5 Cloud Network существует ограничение: тестируемый советник не может записывать на диск более 4ГБ информации и использовать более 4ГБ оперативной памяти. При превышении лимита агент сети не сможет корректно завершить расчет, и вы не получите результат тестирования. При этом с вас будет удержана оплата за уже затраченное на расчеты время.

Если вам необходимо получать информацию с каждого прохода оптимизации, используйте для этого отправку фреймов без записи на диск. Для того, чтобы не использовать файловые операции в советниках при вычислениях в MQL5 Cloud Network, можно применять такую проверку:

```
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(...);
   ...
  }
```

Функция	Действие
FrameFirst	Переводит указатель чтения фреймов в начало и сбрасывает ранее установленный фильтр
FrameFilter	Устанавливает фильтр чтения фреймов и переводит указатель на начало

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
FrameNext	Читает фрейм и перемещает указатель на следующий
FrameInputs	Получает input-параметры, на которых сформирован фрейм
FrameAdd	Добавляет фрейм с данными
ParameterGetRange	Получает для input-переменной информацию о диапазоне значений и шаге изменения при оптимизации эксперта в тестере стратегий
ParameterSetRange	Устанавливает правила использования input-переменной при оптимизации эксперта в тестере стратегий: значение, шаг изменения, начальное и конечное значения

Смотри также

Статистика тестирования, Информация о запущенной MQL5-программе

© 2000-2025, MetaQuotes Ltd.

FrameFirst

Переводит указатель чтения фреймов оптимизации в начало и сбрасывает установленный фильтр.

```
bool  FrameFirst();
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

FrameFilter

Устанавливает фильтр чтения фреймов и переводит указатель на начало.

```
bool  FrameFilter(
   const string  name,          // публичное имя/метка
   long          id             // публичный id
   );
```

name

[in] Публичная метка фрейма. Если передана пустая строка, то фильтр будет работать только по второму параметру.

id

[in] Публичный идентификатор фрейма. Если значение равно ULONG_MAX, то фильтр будет работать только по первому параметру.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Если в качестве первого параметра передана пустая строка, то фильтр будет работать только по числовому параметру, то есть будут просматриваться все фреймы с указанным id. Если значение второго параметра равно ULONG_MAX, то работает только текстовый фильтр.

Вызов FrameFilter("", ULONG_MAX) эквивалентен вызову FrameFirst(), то есть равнозначен отсутствию фильтра.

Пример из статьи SQLite: нативная работа с базами данных на SQL в MQL5:

```
void               CDatabaseFrames::OnTesterDeinit(void)
  {
//--- возьмем имя советника и время окончания оптимизации
   string filename=MQLInfoString(MQL_PROGRAM_NAME)+" "+TimeToString(TimeCurrent())+".s
   StringReplace(filename, ":", "."); // символ ":" запрещен в названиях файлов
//--- открываем/создаем базу данных в общей папке терминалов
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("DB: ", filename, " opened successful");
//--- создаем таблицу PASSES
   if(!DatabaseExecute(db, "CREATE TABLE PASSES("
                       "PASS               INT PRIMARY KEY NOT NULL,"
                       "TRADES             INT,"
                       "WIN_TRADES         INT,"
```

© 2000-2025, MetaQuotes Ltd.

```
                        "PROFIT             REAL,"
                        "GROSS_PROFIT       REAL,"
                        "GROSS_LOSS         REAL,"
                        "SHARPE_RATIO       REAL,"
                        "PROFIT_FACTOR      REAL,"
                        "RECOVERY_FACTOR    REAL,"
                        "EXPECTED_PAYOFF    REAL,"
                        "ON_TESTER          REAL,"
                        "BL_BALANCE         REAL,"
                        "BL_PROFITFACTOR    REAL,"
                        "BL_EXPECTEDPAYOFF  REAL,"
                        "BL_DD              REAL,"
                        "BL_RECOVERYFACTOR  REAL,"
                        "BL_SHARPE          REAL );"))
     {
      Print("DB: ", filename, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- переменные для чтения фреймов
   string        name;
   ulong         pass;
   long          id;
   double        value;
   double        stats[];
//--- переведем указатель фреймов в начало
   FrameFirst();
   FrameFilter("", STATS_FRAME); // выбираем для работы фреймы с торговой статистикой
//--- переменные для получения статистики из фрейма
   int trades;
   double win_trades_percent;
   double profit, gross_profit, gross_loss;
   double sharpe_ratio, profit_factor, recovery_factor, expected_payoff;
   double ontester_value;                              // пользовательский критерий оп
   double balance;                                     // Balance
   double balance_plus_profitfactor;                   // Balance+ProfitFactor
   double balance_plus_expectedpayoff;                 // Balance+ExpectedPayoff
   double balance_plus_dd;                             // Balance+EquityDrawdown
   double balance_plus_recoveryfactor;                 // Balance+RecoveryFactor
   double balance_plus_sharpe;                         // Balance+Sharpe
//--- блокируем базу данных на время массовых транзакций
   DatabaseTransactionBegin(db);
//--- проходим по фреймам и читаем данные из них
   bool failed=false;
   while(FrameNext(pass, name, id, value, stats))
     {
      //Print("Got pass #", pass);    // Раскомментировать только для отладки
      trades=(int)stats[0];
      win_trades_percent=stats[1];
```

© 2000-2025, MetaQuotes Ltd.

```
      profit=stats[2];
      gross_profit=stats[3];
      gross_loss=stats[4];
      sharpe_ratio=stats[5];
      profit_factor=stats[6];
      recovery_factor=stats[7];
      expected_payoff=stats[8];
      ontester_value=stats[9];
      balance=stats[10];
      balance_plus_profitfactor=stats[11];
      balance_plus_expectedpayoff=stats[12];
      balance_plus_dd=stats[13];
      balance_plus_recoveryfactor=stats[14];
      balance_plus_sharpe=stats[15];
      /*  Раскомментировать только для отладки
      PrintFormat("VALUES (%d,%d,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%G,%.2f,%.2f,%2.f,
                  pass, trades, win_trades_percent, profit, gross_profit, gross_loss,
                  profit_factor, recovery_factor, expected_payoff, ontester_value, bal
                  balance_plus_profitfactor, balance_plus_expectedpayoff, balance_plus
                  balance_plus_sharpe);
      */
      //--- запишем данные в таблицу
      string request=StringFormat("INSERT INTO PASSES (PASS,TRADES,WIN_TRADES, PROFIT,
                                  "SHARPE_RATIO,PROFIT_FACTOR,RECOVERY_FACTOR,EXPECTEI
                                  "BL_BALANCE,BL_PROFITFACTOR,BL_EXPECTEDPAYOFF,BL_DD,
                                  "VALUES (%d, %d, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f,
                                  pass, trades, win_trades_percent, profit, gross_prof
                                  profit_factor, recovery_factor, expected_payoff, ont
                                  balance_plus_profitfactor, balance_plus_expectedpayo
                                  balance_plus_sharpe);

      //--- выполним запрос на добавление прохода в таблицу PASSES
      if(!DatabaseExecute(db, request))
        {
         PrintFormat("Failed to insert pass %d with code %d", pass, GetLastError());
         failed=true;
         break;
        }
     }
//--- если при выполнении транзакции произошла ошибка, сообщим и завершим работу
   if(failed)
     {
      Print("Transaction failed, error code=", GetLastError());
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      DatabaseTransactionCommit(db);
      Print("Transaction done successful");
     }
//--- закроем базу данных
   if(db!=INVALID_HANDLE)
     {
      PrintFormat("Close database with handle=%d", db);
      PrintFormat("Database stored in file '%s'", filename);
      DatabaseClose(db);
     }
//---
  }
```

Смотри также

FrameAdd

© 2000-2025, MetaQuotes Ltd.

FrameNext

Читает текущий фрейм и перемещает указатель на следующий. Существует 2 варианта функции.

1. Вызов с получением одного числового значения

```
bool  FrameNext(
   ulong&   pass,      // номер прохода в оптимизации, на котором добавлен фрейм
   string&  name,      // публичное имя/метка
   long&    id,        // публичный id
   double&  value      // значение
   );
```

2. Вызов с получением всех данных фрейма

```
bool  FrameNext(
   ulong&   pass,      // номер прохода в оптимизации, на котором добавлен фрейм
   string&  name,      // публичное имя/метка
   long&    id,        // публичный id
   double&  value,     // значение
   void&    data[]     // массив любого типа
   );
```

Параметры

pass

[out] Номер прохода при оптимизации в тестере стратегий.

name

[out] Имя идентификатора.

id

[out] Значение идентификатора.

value

[out] Одиночное числовое значение.

data

[out] Массив любого типа.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

При использовании второго варианта вызова необходимо правильно обрабатывать полученные данные в массиве *data[]*.

© 2000-2025, MetaQuotes Ltd.

FrameInputs

Получает input-параметры, на которых сформирован фрейм с заданным номером прохода.

```
bool  FrameInputs(
   ulong    pass,                // номер прохода в оптимизации
   string&  parameters[],        // массив строк вида "parameterN=valueN"
   uint&    parameters_count     // общее количество параметров
   );
```

Параметры

pass

[in] Номер прохода при оптимизации в тестере стратегий.

parameters

[out] Строковый массив с описанием имен и значений параметров

parameters_count

[out] Количество элементов в массиве *parameters[]*.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Получив количество строк *parameters_count* в массиве *parameters[]*, можно организовать цикл для перебора всех записей. Это позволит узнать для заданного номера прохода значения входных параметров эксперта.

© 2000-2025, MetaQuotes Ltd.

FrameAdd

Добавляет фрейм с данными. Существует 2 варианта функции.

1. Добавление данных из файла

```
bool  FrameAdd(
   const string  name,        // публичное имя/метка
   long          id,          // публичный id
   double        value,       // значение
   const string  filename     // имя файла с данными
   );
```

2. Добавление данных из массива любого типа

```
bool  FrameAdd(
   const string  name,        // публичное имя/метка
   long          id,          // публичный id
   double        value,       // значение
   const void&   data[]       // массив любого типа
   );
```

Параметры

name

[in] Публичная метка фрейм. Может использоваться для фильтра в функции FrameFilter().

id

[in] Публичный идентификатор фрейма. Может использоваться для фильтра в функции FrameFilter().

value

[in] Числовое значение для записи во фрейм. Предназначен для передачи одиночного результата прохода как в функции OnTester().

filename

[in] Имя файла, в котором находятся данные для добавления во фрейм. Файл должен находиться в папке MQL5/Files.

data

[in] Массив любого типа для записи во фрейм. Передается по ссылке.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

ParameterGetRange

Получает для input-переменной информацию о диапазоне значений и шаге изменения при оптимизации эксперта в тестере стратегий. Существует 2 варианта функции.

1. Получение информации для input-параметра целого типа

```
bool  ParameterGetRange(
   const string  name,           // имя параметра (input-переменной)
   bool&         enable,         // разрешена оптимизация параметра
   long&         value,          // значение параметра
   long&         start,          // начальное значение
   long&         step,           // шаг изменения
   long&         stop            // конечное значение
   );
```

2. Получение информации для input-параметра вещественного типа

```
bool  ParameterGetRange(
   const string  name,           // имя параметра (input-переменной)
   bool&         enable,         // разрешена оптимизация параметра
   double&       value,          // значение параметра
   double&       start,          // начальное значение
   double&       step,           // шаг изменения
   double&       stop            // конечное значение
   );
```

Параметры

name

[in] Идентификатор input-переменной. Такие переменные являются внешними параметрами программы, значения которых можно задавать при запуске на графике или при одиночном тестировании.

enable

[out] Признак того, что данный параметр можно использовать для перебора значений в процессе оптимизации в тестере стратегий.

value

[out] Значение параметра.

start

[out] Начальное значение параметра при оптимизации.

step

[out] Шаг изменения параметра при переборе его значений.

stop

[out] Конечное значение параметра при оптимизации.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успешного выполнения, иначе false. Для получения информации об ошибке используйте функцию GetLastError().

Примечание

Функция может вызываться только из обработчиков OnTesterInit(), OnTesterPass() и OnTesterDeinit(). Предназначена для получения значения и диапазона изменения входных параметров эксперта в процессе оптимизации в тестере стратегий.

При вызове в OnTesterInit() полученную информацию можно использовать для переопределения правила перебора любой input-переменной с помощью функции ParameterSetRange(). Таким образом можно задавать новые значения Start, Stop, Step и даже полностью исключать данный параметр из оптимизации несмотря на настройки в тестере стратегий. Это позволяет создавать собственные сценарии для управления пространством входных параметров при оптимизации, то есть исключать из оптимизации одни параметры в зависимости от значений ключевых параметров эксперта.

Пример:

```
#property description "Эксперт для демонстрации функции ParameterGetRange()."
#property description "Необходимо запускать в тестере стратегий в режиме оптимизации"
//--- input parameters
input int                  Input1=1;
input double               Input2=2.0;
input bool                 Input3=false;
input ENUM_DAY_OF_WEEK     Input4=SUNDAY;

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- эксперт предназначен только для работы в тестере стратегий
   if(!MQL5InfoInteger(MQL5_OPTIMIZATION))
     {
      MessageBox("Необходимо запускать в тестере стратегий в режиме оптимизации!");
      //--- завершаем работу эксперта досрочно и снимаем с графика
      return(INIT_FAILED);
     }
//--- успешное завершение инициализации
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| TesterInit function                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- пример для input-параметра типа long
   string name="Input1";
   bool enable;
   long par1,par1_start,par1_step,par1_stop;
```

© 2000-2025, MetaQuotes Ltd.

```
   ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
   Print("Первый параметр");
   PrintFormat("%s=%d  enable=%s  from %d to %d with step=%d",
               name,par1,(string)enable,par1_start,par1_stop,par1_step);
//--- пример для input-параметра типа double
   name="Input2";
   double par2,par2_start,par2_step,par2_stop;
   ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
   Print("Второй параметр");
   PrintFormat("%s=%G  enable=%s  from %G to %G with step=%G",
               name,par2,(string)enable,par2_start,par2_stop,par2_step);

//--- пример для input-параметра типа bool
   name="Input3";
   long par3,par3_start,par3_step,par3_stop;
   ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
   Print("Третий параметр");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,(string)par3,(string)enable,
               (string)par3_start,(string)par3_stop);
//--- пример для input-параметра типа перечисление
   name="Input4";
   long par4,par4_start,par4_step,par4_stop;
   ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
   Print("Четвертый параметр");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
               EnumToString((ENUM_DAY_OF_WEEK)par4_start),
               EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
  }
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- это сообщение выйдет по окончании оптимизации
   Print(__FUNCTION__," Optimization completed");
  }
```

© 2000-2025, MetaQuotes Ltd.

ParameterSetRange

Устанавливает правила использования input-переменной при оптимизации эксперта в тестере стратегий: значение, шаг изменения, начальное и конечное значения. Существует 2 варианта функции.

1. Установка значений для input-параметра целого типа

```
bool  ParameterSetRange(
   const string  name,          // имя параметра (input-переменной)
   bool          enable,        // разрешить оптимизацию параметра
   long          value,         // значение параметра
   long          start,         // начальное значение
   long          step,          // шаг изменения
   long          stop           // конечное значение
   );
```

2. Установка значений для input-параметра вещественного типа

```
bool  ParameterSetRange(
   const string  name,          // имя параметра (input-переменной)
   bool          enable,        // разрешить оптимизацию параметра
   double        value,         // значение параметра
   double        start,         // начальное значение
   double        step,          // шаг изменения
   double        stop           // конечное значение
   );
```

Параметры

name

[in] Идентификатор переменной input или sinput. Такие переменные являются внешними параметрами программы, значения которых можно задавать при запуске.

enable

[in] Разрешить данный параметр для перебора значений в процессе оптимизации в тестере стратегий.

value

[in] Значение параметра.

start

[in] Начальное значение параметра при оптимизации.

step

[in] Шаг изменения параметра при переборе его значений.

stop

[in] Конечное значение параметра при оптимизации.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает true в случае успешного выполнения, иначе false. Для получения информации об ошибке используйте функцию GetLastError().

Примечание

Функция может вызываться только из обработчика OnTesterInit() при запуске оптимизации в тестере стратегий. Предназначена для установки диапазона и шага изменения параметра, в том числе позволяет полностью исключать этот параметр из процедуры оптимизации, несмотря на настройки в тестере стратегий. Также позволяет использовать в оптимизации даже переменные, объявленные с модификатором sinput.

Функция ParameterSetRange() позволяет создавать собственные сценарии оптимизации эксперта в тестере стратегий в зависимости от значений его ключевых параметров, то есть включать либо исключать из оптимизации требуемые входные параметры, а также задавать требуемые диапазон и шаг изменения.

© 2000-2025, MetaQuotes Ltd.

Работа с событиями

Функции для работы с пользовательскими событиями и событиями таймера. Кроме этих функций существуют также специальные функции, для обработки предопределенных событий.

Функция	Действие
EventSetMillisecondTimer	Запускает генератор событий таймера высокого разрешения с периодом менее 1 секунды для текущего графика
EventSetTimer	Запускает генератор событий таймера с указанной периодичностью для текущего графика
EventKillTimer	Останавливает на текущем графике генерацию событий по таймеру
EventChartCustom	Генерирует пользовательское событие для указанного графика

Смотри также

Типы событий графика

© 2000-2025, MetaQuotes Ltd.

EventSetMillisecondTimer

Указывает клиентскому терминалу, что для данного эксперта или индикатора необходимо генерировать события таймера с периодичностью менее одной секунды.

```
bool  EventSetMillisecondTimer(
   int  milliseconds      // количество миллисекунд
   );
```

Параметры

milliseconds

[in] Количество миллисекунд, определяющее периодичность возникновения событий от таймера.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе - false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Эта функция предназначена для тех случаев, когда требуется таймер высокого разрешения, то есть нужно получать события таймера чаще, чем один раз в секунду. Если вам достаточно обычного таймера с периодом более 1 секунды, то используйте EventSetTimer().

В общем случае при уменьшении периода таймера увеличивается время тестирования, так как возрастает количество вызовов обработчика событий таймера. При работе в режиме реального времени события таймера генерируются не чаще 1 раза в 10-16 миллисекунд, что связано с аппаратными ограничениями.

Обычно эта функция должна вызываться из функции OnInit() или в конструкторе класса. Для того чтобы обрабатывать события, приходящие от таймера, эксперт или индикатор должен иметь функцию OnTimer().

Каждый эксперт и каждый индикатор работает со своим таймером и получает события только от него. При завершении работы mql5-программы таймер уничтожается принудительно, если он был создан, но не был отключен функцией EventKillTimer().

Для каждой программы может быть запущено не более одного таймера. Каждая mql5-программа и каждый график имеют свою собственную очередь событий, куда складываются все вновь поступающие события. Если в очереди уже есть событие Timer либо это событие находится в состоянии обработки, то новое событие Timer в очередь mql5-программы не ставится.

© 2000-2025, MetaQuotes Ltd.

EventSetTimer

Указывает клиентскому терминалу, что для данного эксперта или индикатора необходимо генерировать события от таймера с указанной периодичностью.

```
bool  EventSetTimer(
   int  seconds      // количество секунд
   );
```

Параметры

seconds

[in] Количество секунд, определяющее периодичность возникновения событий от таймера.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Обычно, эта функция должна вызываться из функции OnInit() или в конструкторе класса. Для того чтобы обрабатывать события, приходящие от таймера, эксперт или индикатор должен иметь функцию OnTimer().

Каждый эксперт и каждый индикатор работает со своим таймером, и получает события только от него. При завершении работы mql5-программы таймер уничтожается принудительно, если он был создан, но не был отключен функцией EventKillTimer().

Для каждой программы может быть запущено не более одного таймера. Каждая mql5-программа и каждый график имеют свою собственную очередь событий, куда складываются все вновь поступающие события. Если в очереди уже есть событие Timer либо это событие находится в состоянии обработки, то новое событие Timer в очередь mql5-программы не ставится.

© 2000-2025, MetaQuotes Ltd.

EventKillTimer

Указывает клиентскому терминалу, что необходимо остановить генерацию событий от таймера для данного эксперта или индикатора.

```
void  EventKillTimer();
```

Возвращаемое значение

Нет возвращаемого значения.

Примечание

Обычно эта функция должна вызываться из функции OnDeinit() в том случае, если в функции OnInit() была вызвана функция EventSetTimer(). Либо должна вызываться из деструктора класса, если в конструкторе этого класса вызывается функция EventSetTimer().

Каждый эксперт и каждый индикатор работает со своим таймером, и получает события только от него. При завершении работы mql5-программы таймер уничтожается принудительно, если он был создан, но не был отключен функцией EventKillTimer().

© 2000-2025, MetaQuotes Ltd.

EventChartCustom

Генерирует пользовательское событие для указанного графика.

```
bool  EventChartCustom(
   long    chart_id,            // идентификатор графика-получателя события
   ushort  custom_event_id,     // идентификатор события
   long    lparam,              // параметр типа long
   double  dparam,              // параметр типа double
   string  sparam               // строковый параметр события
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

custom_event_id

[in] Идентификатор пользовательского события. Этот идентификатор автоматически добавляется к значению CHARTEVENT_CUSTOM и преобразуется к целому типу.

lparam

[in] Параметр события типа long, передаваемый функции OnChartEvent.

dparam

[in] Параметр события типа double, передаваемый функции OnChartEvent.

sparam

[in] Параметр события типа string, передаваемый функции OnChartEvent. Если строка имеет длину больше, чем 63 символа, то строка усекается.

Возвращаемое значение

Возвращает true в случае удачной постановки пользовательского события в очередь событий графика - получателя события. В случае ошибки возвращает false, для получения кода ошибки используйте GetLastError().

Примечание

Эксперт или индикатор, прикрепленный к указанному графику, обрабатывает данное событие при помощи функции OnChartEvent(int event_id, long& lparam, double& dparam, string& sparam).

Для каждого типа события входные параметры функции OnChartEvent() имеют определенные значения, которые необходимы для обработки этого события. В таблице перечислены события и значения, которые передаются через параметры.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
Событие нажатия клавиатуры	CHARTEVENT_KEYDOWN	код нажатой клавиши	Количество нажатий клавиши,сгенер	Строковое значение битовой маски,

© 2000-2025, MetaQuotes Ltd.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
			ированных за время её удержания в нажатом состоянии	описывающее статус кнопок клавиатуры
События мыши (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	X координата	Y координата	Строковое значение битовой маски, описывающее статус кнопок мыши
Событие создания графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Имя созданного графического объекта
Событие изменения свойств объекта через диалог свойств	CHARTEVENT_OBJECT_CHANGE	—	—	Имя измененного графического объекта
Событие удаления графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_DELETE=true)	CHARTEVENT_OBJECT_DELETE	—	—	Имя удаленного графического объекта
Событие щелчка мыши на графике	CHARTEVENT_CLICK	X координата	Y координата	—
Событие щелчка мыши на	CHARTEVENT_OBJECT_CLICK	X координата	Y координата	Имя графического

© 2000-2025, MetaQuotes Ltd.

Событие	Значение параметра id	Значение параметра lparam	Значение параметра dparam	Значение параметра sparam
графическом объекте				объекта, на котором произошло событие
Событие перемещения графического объекта при помощи мыши	CHARTEVENT_OBJECT_DRAG	—	—	Имя перемещенного графического объекта
Событие окончания редактирования текста в поле ввода графического объекта "Поле ввода"	CHARTEVENT_OBJECT_ENDEDIT	—	—	Имя графического объекта "Поле ввода", в котором завершилось редактирование текста
Событие изменения графика	CHARTEVENT_CHART_CHANGE	—	—	—
Пользовательское событие с номером N	CHARTEVENT_CUSTOM+N	Значение, заданное функцией EventChartCustom()	Значение, заданное функцией EventChartCustom()	Значение, заданное функцией EventChartCustom()

Пример:

```
//+------------------------------------------------------------------+
//|                                            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="Button";
string labelID="Info";
int broadcastEventID=5000;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- создадим кнопку, для передачи пользовательских событий
   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,"Кнопка");
   ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
   ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- создадим метку для вывода информации
   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,"Нет информации");
   ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
   ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ObjectDelete(0,buttonID);
   ObjectDelete(0,labelID);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- проверим событие на нажатие кнопки мышки
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      string clickedChartObject=sparam;
      //--- если нажатие на объекте с именем buttonID
      if(clickedChartObject==buttonID)
        {
         //--- состояние кнопки - нажата кнопка или нет
         bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
         //--- выведем в лог отладочное сообщение
         Print("Кнопка нажата = ",selected);
         int customEventID; // номер пользовательского события для отправки
         string message;    // сообщение для отправки в событии
         //--- если кнопка нажата
         if(selected)
           {
            message="Кнопка нажата";
            customEventID=CHARTEVENT_CUSTOM+1;
           }
         else // кнопка не нажата
           {
            message="Кнопка отжата";
            customEventID=CHARTEVENT_CUSTOM+999;
           }
         //--- отправим пользовательское событие "своему"графику
         EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
         //--- отправим сообщение всем открытым графикам
         BroadcastEvent(ChartID(),0,"Broadcast Message");
         //--- отладочное сообщение
         Print("Отправлено событие с ID = ",customEventID);
        }
      ChartRedraw();// принудительно перерисуем все объекты на графике
     }

//--- проверим событие на принадлежность к пользовательским событиям
   if(id>CHARTEVENT_CUSTOM)
     {
      if(id==broadcastEventID)
        {
         Print("Получили широковещательное сообщение от графика с id = "+lparam);
        }
      else
        {
         //--- прочитаем текстовое сообщение в событии
         string info=sparam;
         Print("Обрабатывается ПОЛЬЗОВАТЕЛЬСКОЕ событие с ID = ",id);
         //--- выведем сообщение в метке
         ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
         ChartRedraw();// принудительно перерисуем все объекты на графике
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      }
   }
//+------------------------------------------------------------------+
//| послать широковещательное сообщение всем открытым графикам       |
//+------------------------------------------------------------------+
void BroadcastEvent(long lparam,double dparam,string sparam)
   {
    int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
    long currChart=ChartFirst();
    int i=0;
    while(i<CHARTS_MAX)                // у нас наверняка не больше CHARTS_MAX открытых
      {
       EventChartCustom(currChart,eventID,lparam,dparam,sparam);
       currChart=ChartNext(currChart); // на основании предыдущего получим новый графи
       if(currChart==-1) break;        // достигли конца списка графиков
       i++;                            // не забудем увеличить счетчик
      }
   }
//+------------------------------------------------------------------+
```

Смотри также

События клиентского терминала, Функции обработки событий

© 2000-2025, MetaQuotes Ltd.

Работа с OpenCL

Программы на OpenCL предназначены для выполнения вычислений на видеокартах с поддержкой стандарта OpenCL 1.1 или выше. Современные видеокарты содержат сотни небольших специализированных процессоров, которые могут одновременно выполнять простые математические операции над входящими потоками данных. Язык OpenCL берёт на себя организацию таких параллельных вычислений и позволяет добиться огромного ускорения для некоторого класса задач.

На некоторых видеокартах по умолчанию отключен режим работы с числами типа double, что приводит к возникновению ошибки компиляции 5105. Для включения режима поддержки чисел double в текст OpenCL-программы нужно добавить директиву #pragma OPENCL EXTENSION cl_khr_fp64 : enable. Однако, если видеокарта не поддерживает double, то включение данной директивы не поможет.

Исходный код OpenCL рекомендуется писать в отдельных CL-файлах, которые затем можно подключать к MQL5-программе с помощью ресурсных переменных.

Обработка ошибок в OpenCL программах

Для получения информации о последней ошибке в OpenCL-программе необходимо использовать функции CLGetInfoInteger и CLGetInfoString, которые позволяют получить код ошибки и текстовое описание.

Код последней ошибки OpenCL: Для получения последней ошибки OpenCL используйте вызов CLGetInfoInteger, при этом параметр *handle* игнорируется, то есть не имеет значения (можно указать ноль). Описание ошибок: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS.

Для неизвестного кода ошибки вернётся строка "unknown OpenCL error N", где вместо N будет код ошибки. Пример:

```
//--- первый параметр handle игнорируется при получении кода последней ошибки
int code = (int)CLGetInfoInteger(0,CL_LAST_ERROR);
```

Текстовое описание ошибки OpenCL: Для получения последней ошибки OpenCL используйте вызов CLGetInfoString, при этом в качестве параметра *handle* следует передать код ошибки.

Описание ошибок: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Если вместо кода ошибки передать CL_LAST_ERROR, то функция вернёт описание последней ошибки. Пример:

```
//--- получить код последней OpenCL ошибки
int    code = (int)CLGetInfoInteger(0,CL_LAST_ERROR);
string desc; // для получения текстового описания ошибки

//--- используем код ошибки для получения текстового описания ошибки
if(!CLGetInfoString(code,CL_ERROR_DESCRIPTION,desc))
   desc = "cannot get OpenCL error description, " + (string)GetLastError();
Print(desc);

//--- чтобы получить описание последней OpenCL ошибки без предварительного получения к
```

© 2000-2025, MetaQuotes Ltd.

```
if(!CLGetInfoString(CL_LAST_ERROR,CL_ERROR_DESCRIPTION, desc))
   desc = "cannot get OpenCL error description, " + (string)GetLastError();
Print(desc);
```

Пока в качестве описания ошибки отдаётся имя внутреннего перечисления, расшифровку которого можно посмотреть на странице https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Например, если получено значение CL_INVALID_KERNEL_ARGS, описание для него "Returned when enqueuing a kernel when some kernel arguments have not been set or are invalid."

Функции для выполнения программ на OpenCL:

Функция	Действие
CLHandleType	Возвращает тип OpenCL хендла в виде значения из перечисления ENUM_OPENCL_HANDLE_TYPE
CLGetInfoInteger	Возвращает значение целочисленного свойства для OpenCL-объекта или устройства
CLGetInfoString	Возвращает строковое значение свойства для OpenCL-объекта или устройства
CLContextCreate	Создает контекст OpenCL
CLContextFree	Удаляет контекст OpenCL
CLGetDeviceInfo	Получает свойство устройства из OpenCL драйвера
CLProgramCreate	Создает OpenCL программу из исходного кода
CLProgramFree	Удаляет OpenCL программу
CLKernelCreate	Создает функцию запуска OpenCL
CLKernelFree	Удаляет функцию запуска OpenCL
CLSetKernelArg	Выставляет параметр для функции OpenCL
CLSetKernelArgMem	Выставляет буфер OpenCL в качестве параметра функции OpenCL
CLSetKernelArgMemLocal	Задаёт локальный буфер в качестве аргумента kernel-функции
CLBufferCreate	Создает буфер OpenCL
CLBufferFree	Удаляет буфер OpenCL
CLBufferWrite	Записывает массив в буфер OpenCL и возвращает количество записанных элементов
CLBufferRead	Читает буфер OpenCL в массив и возвращает количество прочитанных элементов
CLExecute	Выполняет OpenCL программу
CLExecutionStatus	Возвращает состояние выполнения OpenCL программы

© 2000-2025, MetaQuotes Ltd.

Смотри также

OpenCL, Ресурсы

© 2000-2025, MetaQuotes Ltd.

CLHandleType

Возвращает тип OpenCL хендла в виде значения из перечисления ENUM_OPENCL_HANDLE_TYPE.

```
ENUM_OPENCL_HANDLE_TYPE  CLHandleType(
   int  handle     // хендл объекта OpenCL
   );
```

Параметры

handle

[in] Хендл на объект OpenCL: контекст, кернел, буфер или программа OpenCL.

Возвращаемое значение

Тип хендла OpenCL в виде значения из перечисления ENUM_OPENCL_HANDLE_TYPE.

ENUM_OPENCL_HANDLE_TYPE

Идентификатор	Описание
OPENCL_INVALID	Некорректный хендл
OPENCL_CONTEXT	Хендл контекста OpenCL
OPENCL_PROGRAM	Хендл программы OpenCL
OPENCL_KERNEL	Хендл кернела OpenCL
OPENCL_BUFFER	Хендл буфера OpenCL

© 2000-2025, MetaQuotes Ltd.

CLGetInfoInteger

Возвращает значение целочисленного свойства для OpenCL-объекта или устройства.

```
long  CLGetInfoInteger(
   int  handle,                          // хендл объекта OpenCL или номер OpenCL уст
   ENUM_OPENCL_PROPERTY_INTEGER  prop    // запрашиваемое свойство
   );
```

Параметры

handle

[in] Хендл на объект OpenCL или номер OpenCL устройства. Нумерация OpenCL устройств начинается с нуля.

prop

[in] Тип запрашиваемого свойства из перечисления ENUM_OPENCL_PROPERTY_INTEGER, значение которого нужно получить.

Возвращаемое значение

Значение указанного свойства при успешном выполнении или -1 в случае ошибки. Для получения информации об ошибке используйте функцию GetLastError().

ENUM_OPENCL_PROPERTY_INTEGER

Идентификатор	Описание	Тип
CL_DEVICE_COUNT	Количество устройств с поддержкой OpenCL. Для этого свойства не требуется указание первого параметра, то есть можно передать нулевое значение для параметра *handle*.	int
CL_DEVICE_TYPE	Тип устройства	ENUM_CL_DEVICE_TYPE
CL_DEVICE_VENDOR_ID	Уникальный идентификатор производителя	uint
CL_DEVICE_MAX_COMPUTE_UNITS	Количество параллельных вычисляемых задач в устройстве OpenCL. Одна рабочая группа выполняет одну вычислительную задачу. Минимальное значение равно 1	uint
CL_DEVICE_MAX_CLOCK_FREQUENCY	Максимальная установленная частота устройства в МГц.	uint
CL_DEVICE_GLOBAL_MEM_SIZE	Размер глобальной памяти устройства в байтах	ulong

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание	Тип
CL_DEVICE_LOCAL_MEM_SIZE	Размер локальной памяти обрабатываемых данных (сцены) в байтах	uint
CL_BUFFER_SIZE	Реальный размер буфера OpenCL в байтах	ulong
CL_DEVICE_MAX_WORK_GROUP_SIZE	Общее количество локальных рабочих групп доступных для OpenCL устройства.	ulong
CL_KERNEL_WORK_GROUP_SIZE	Общее количество локальных рабочих групп, доступных для OpenCL программы.	ulong
CL_KERNEL_LOCAL_MEM_SIZE	Размер локальной памяти в байтах, которую использует OpenCL программа для всех параллельных задач в группе. Используйте CL_DEVICE_LOCAL_MEM_SIZE для получения доступного максимума	ulong
CL_KERNEL_PRIVATE_MEM_SIZE	Минимальный размер приватной памяти в байтах, используемый каждой задачей в кернеле OpenCL программы.	ulong
CL_LAST_ERROR	Значение последней OpenCL ошибки	int

Перечисление ENUM_CL_DEVICE_TYPE содержит возможные типы устройств с поддержкой OpenCL. Получить тип устройства можно по его номеру или хэндлу объекта OpenCL с помощью вызова CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE).

ENUM_CL_DEVICE_TYPE

Идентификатор	Описание
CL_DEVICE_ACCELERATOR	Специализированный OpenCL ускоритель (например, IBM CELL Blade).
CL_DEVICE_CPU	Использование центрального процессора компьютера в качестве OpenCL устройства. Центральный процессор может иметь одно или более вычислительных ядер.
CL_DEVICE_GPU	OpenCL устройство на основе видеокарты.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
CL_DEVICE_DEFAULT	OpenCL устройство по умолчанию. Устройством по умолчанию не может являться CL_DEVICE_TYPE_CUSTOM устройство.
CL_DEVICE_CUSTOM	Специализированные ускорители, которые не поддерживают программы на OpenCL C.

Пример:

```
void OnStart()
  {
   int cl_ctx;
//--- инициализация OpenCL контекста
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
//--- Выведем общую информацию об устройстве OpenCL
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY)," MHz"
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE)," by
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE)," byte
//--- освободим OpenCL контекстс
   CLContextFree(cl_ctx);
  }
```

© 2000-2025, MetaQuotes Ltd.

CLGetInfoString

Возвращает строковое значение свойства для OpenCL-объекта или устройства.

```
bool  CLGetInfoString(
   int  handle,                          // хендл объекта OpenCL или номер OpenCL уст
   ENUM_OPENCL_PROPERTY_STRING  prop,    // запрашиваемое свойство
   string&  value                        // строка по ссылке
   );
```

Параметры

handle

[in] Хендл на объект OpenCL или номер OpenCL устройства. Нумерация OpenCL устройств начинается с нуля.

prop

[in] Тип запрашиваемого свойства из перечисления ENUM_OPENCL_PROPERTY_STRING, значение которого нужно получить.

value

[out] Строка для получения значения свойства.

Возвращаемое значение

true при успешном выполнении или false в случае ошибки. Для получения информации об ошибке используйте функцию GetLastError().

ENUM_OPENCL_PROPERTY_STRING

Идентификатор	Описание
CL_PLATFORM_PROFILE	CL_PLATFORM_PROFILE - Профиль OpenCL. Имя профиля может быть одним из значений: • FULL_PROFILE - реализация поддерживает OpenCL (функциональность определена как часть спецификации ядра и не требует дополнительных расширений для поддержки OpenCL); • EMBEDDED_PROFILE - реализация поддерживает OpenCL в виде дополнения. Дополненный профиль определяется как подмножество для каждой версии OpenCL.
CL_PLATFORM_VERSION	Версия OpenCL
CL_PLATFORM_VENDOR	Название производителя устройства
CL_PLATFORM_EXTENSIONS	Список расширений, поддерживаемых платформой. Названия расширений должны поддерживаться всеми устройствами, связанными с данной платформой

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
CL_DEVICE_NAME	Название устройства
CL_DEVICE_VENDOR	Название производителя
CL_DRIVER_VERSION	Версия драйвера OpenCL в формате major_number.minor_number
CL_DEVICE_PROFILE	Профиль OpenCL устройства. Имя профиля может быть одним из следующих значений: • FULL_PROFILE - реализация поддерживает OpenCL (функциональность определена как часть спецификации ядра и не требует дополнительных расширений для поддержки OpenCL); • EMBEDDED_PROFILE - реализация поддерживает OpenCL в виде дополнения. Дополненный профиль определяется как подмножество для каждой версии OpenCL.
CL_DEVICE_VERSION	Версия OpenCL в формате "OpenCL<space><major_version.minor_version><space><vendor-specific information>"
CL_DEVICE_EXTENSIONS	Список расширений, поддерживаемых устройством. Список может включать расширения, поддерживаемые производителем и содержать один или более одобренных имен: 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	Список кернелов, поддерживаемых устройством, разделенных ";" .
CL_DEVICE_OPENCL_C_VERSION	Максимальная версия, поддерживаемая компилятором для данного устройства. Формат версии: "OpenCL<space>C<space><major_version.minor_version><space><vendor-specific information> "
CL_ERROR_DESCRIPTION	Текстовое описание ошибки OpenCL

Пример:

```
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   int cl_ctx;
   string str;
//--- инициализация OpenCL контекста
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
//--- Выведем информацию о платформе
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
      Print("OpenCL platform name: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
      Print("OpenCL platform vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
      Print("OpenCL platform ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
      Print("OpenCL platform profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
      Print("OpenCL platform ext: ",str);
//--- Выведем информацию о самом устройстве
   if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
      Print("OpenCL device name: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
      Print("OpenCL device profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
      Print("OpenCL device kernels: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
      Print("OpenCL device ext: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
      Print("OpenCL device vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
      Print("OpenCL device ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
      Print("OpenCL open c ver: ",str);
//--- Выведем общую информацию об устройстве OpenCL
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY));
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE));
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
//--- освободим OpenCL контекст
   CLContextFree(cl_ctx);

  }
```

© 2000-2025, MetaQuotes Ltd.

CLContextCreate

Создает контекст OpenCL и возвращает хендл на него.

```
int  CLContextCreate(
   int  device=CL_USE_ANY     // порядковый номер OpenCL устройства или макрос
   );
```

Параметры

device

[in] Номер OpenCL-устройства в системе по порядку. Вместо конкретного номера можно указать одно из значений:

- CL_USE_ANY - разрешается использовать любое доступное устройство с поддержкой OpenCL;
- CL_USE_CPU_ONLY - разрешается использовать только эмуляцию OpenCL на CPU;
- CL_USE_GPU_ONLY - эмуляция OpenCL запрещена и разрешается использовать только специализированные устройства с поддержкой OpenCL (видеокарты);
- CL_USE_GPU_DOUBLE_ONLY - разрешается использовать только GPU, которые поддерживают тип double.

Возвращаемое значение

Хендл на контекст OpenCL при успешном создании, или -1 в случае ошибки. Для получения информации об ошибке используйте функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLContextFree

Удаляет контекст OpenCL.

```
void  CLContextFree(
   int  context     // хендл на контекст OpenCL
   );
```

Параметры

context

[in] Хендл контекста OpenCL.

Возвращаемое значение

Нет. В случае внутренней ошибки изменяется значение _LastError. Для получения информации об ошибке используйте функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLGetDeviceInfo

Получает свойство устройства из OpenCL драйвера.

```
bool  CLGetDeviceInfo(
   int     handle,          // хендл устройства OpenCL
   int     property_id,     // идентификатор запрашиваемого свойства
   uchar&  data[],          // массив для получения данных
   uint&   size             // смещение в массиве в элементах, по умолчанию 0
   );
```

Параметры

handle

[in] Номер OpenCL устройства или хендл OpenCL, созданный функцией CLContextCreate().

property_id

[in] Идентификатор свойства, которое необходимо получить об устройстве OpenCL. Может быть одним из предопределенных значений, перечисленных в таблице ниже.

data[]

[out] Массив для получения данных о запрашиваемом свойстве.

size

[out] Размер полученных данных в массиве *data[]*.

Возвращаемое значение

true при успешном выполнении или false в случае ошибки. Для получения информации об ошибке используйте функцию GetLastError().

Примечание

Для одномерных массивов номер элемента, с которого начинается чтение данных для записи в буфер OpenCL, вычисляется с учётом флага AS_SERIES.

Перечень допустимых идентификаторов свойств OpenCL устройства

Точное описание свойства и его назначение можно найти на официальном сайте OpenCL.

Идентификатор	Значение
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
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR	0x1006

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение
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
CL_DEVICE_MAX_CONSTANT_ARGS	0x1021
CL_DEVICE_LOCAL_MEM_TYPE	0x1022

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение
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
CL_DEVICE_IMAGE_MAX_BUFFER_SIZE	0x1040
CL_DEVICE_IMAGE_MAX_ARRAY_SIZE	0x1041

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение
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

Пример:

```
void OnStart()
  {
//---
   int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
   for(int i = 0; i<dCount; i++)
     {
      int clCtx=CLContextCreate(i);
      if(clCtx == -1)
         Print("ERROR in 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("size = ",size);
      string str=CharArrayToString(data);
      Print(str);
     }
  }
//--- пример вывода в журнал Эксперты
//  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
//  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) i7-377
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      NVIDIA Corporation
```

© 2000-2025, MetaQuotes Ltd.

```
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 19
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      0: GeForce GTX 580
```

© 2000-2025, MetaQuotes Ltd.

CLProgramCreate

Создает OpenCL программу из исходного кода.

```
int  CLProgramCreate(
   int           context,     // хендл на контекст OpenCL
   const string  source       // исходный код
   );
```

Перегруженная версия функции создает OpenCL программу и записывает сообщения компилятора в переданную строку.

```
int  CLProgramCreate(
   int           context,     // хендл на контекст OpenCL
   const string  source,      // исходный код
   string       &build_log    // строка для получения лога компиляции
   );
```

Параметры

context

[in] Хендл контекста OpenCL.

source

[in] Строка с исходным кодом OpenCL программы.

&build_log

[in] Строка для получения сообщений компилятора OpenCL.

Возвращаемое значение

Хендл на объект OpenCL при успешном выполнении. В случае ошибки возвращает -1. Для получения информации об ошибке используйте функцию GetLastError() и выводите значения строки *build_log*.

Примечание

На данный момент предусмотрены следующие коды ошибок:

- ERR_OPENCL_INVALID_HANDLE - невалидный хендл на context OpenCL,
- ERR_INVALID_PARAMETER - невалидный строковой параметр,
- ERR_NOT_ENOUGH_MEMORY - недостаточно памяти для завершения операции,
- ERR_OPENCL_PROGRAM_CREATE - внутренняя ошибка OpenCL или ошибка компиляции.

На некоторых видеокартах по умолчанию отключен режим работы с числами типа double, что приводит к возникновению ошибки компиляции 5105. Для включения режима поддержки чисел double в текст OpenCL-программы нужно добавить директиву #pragma OPENCL EXTENSION cl_khr_fp64 : enable. Однако, если видеокарта не поддерживает double, то включение данной директивы не поможет.

Пример:

```
//+------------------------------------------------------------------+
//| OpenCL kernel                                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
const string
cl_src=
       //--- by default some GPU doesn't support doubles
       //--- cl_khr_fp64 directive is used to enable work with doubles
       "#pragma OPENCL EXTENSION cl_khr_fp64 : enable      \r\n"
       //--- OpenCL kernel function
       "__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"
       "   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)
  {
//--- check array size
   if(ArraySize(data)==0) return(false);
//--- check array index
   if(id>total_arrays) return(false);
//--- calculate local offset for array with index id
   int local_start_offset=id*N;
//--- multiply elements by 2
   for(int i=0; i<N; i++)
     {
      data[i+local_start_offset]*=2.0;
     }
   return true;
  }
//---
#define ARRAY_SIZE   100  // size of the array
#define TOTAL_ARRAYS 5    // total arrays
//--- OpenCL handles
int cl_ctx;  // OpenCL context handle
int cl_prg;  // OpenCL program handle
int cl_krn;  // OpenCL kernel handle
int cl_mem;  // OpenCL buffer handle
//---
double DataArray1[]; // data array for CPU calculation
double DataArray2[]; // data array for GPU calculation
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- initialize OpenCL objects
//--- create OpenCL context
   if((cl_ctx=CLContextCreate())==INVALID_HANDLE)
     {
      Print("OpenCL not found. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL program
   if((cl_prg=CLProgramCreate(cl_ctx,cl_src))==INVALID_HANDLE)
     {
      CLContextFree(cl_ctx);
      Print("OpenCL program create failed. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL kernel
   if((cl_krn=CLKernelCreate(cl_prg,"Test_GPU"))==INVALID_HANDLE)
     {
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL kernel create failed. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL buffer
   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("OpenCL buffer create failed. Error=",GetLastError());
      return(1);
     }
//--- set OpenCL kernel constant parameters
   CLSetKernelArgMem(cl_krn,0,cl_mem);
   CLSetKernelArg(cl_krn,1,ARRAY_SIZE);
   CLSetKernelArg(cl_krn,2,TOTAL_ARRAYS);
//--- prepare data arrays
   ArrayResize(DataArray1,ARRAY_SIZE*TOTAL_ARRAYS);
   ArrayResize(DataArray2,ARRAY_SIZE*TOTAL_ARRAYS);
//--- fill arrays with data
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculate local start offset for jth array
      uint local_offset=j*ARRAY_SIZE;
      //--- prepare array with index j
      for(int i=0; i<ARRAY_SIZE; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- fill arrays with function MathCos(i+j);
          DataArray1[i+local_offset]=MathCos(i+j);
          DataArray2[i+local_offset]=MathCos(i+j);
         }
      };
//--- test CPU calculation
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculation of the array with index j
      Test_CPU(DataArray1,ARRAY_SIZE,j,TOTAL_ARRAYS);
     }
//--- prepare CLExecute params
   uint  offset[]={0};
//--- global work size
   uint  work[]={TOTAL_ARRAYS};
//--- write data to OpenCL buffer
   CLBufferWrite(cl_mem,DataArray2);
//--- execute OpenCL kernel
   CLExecute(cl_krn,1,offset,work);
//--- read data from OpenCL buffer
   CLBufferRead(cl_mem,DataArray2);
//--- total error
   double total_error=0;
//--- compare results and calculate error
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculate local offset for jth array
      uint local_offset=j*ARRAY_SIZE;
      //--- compare the results
      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;
         //--- show first and last 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);
//--- delete OpenCL objects
//--- free OpenCL buffer
   CLBufferFree(cl_mem);
//--- free OpenCL kernel
   CLKernelFree(cl_krn);
//--- free OpenCL program
   CLProgramFree(cl_prg);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- free OpenCL context
   CLContextFree(cl_ctx);
//---
   return(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

CLProgramFree

Удаляет OpenCL программу.

```
void  CLProgramFree(
   int  program     // хендл на объект OpenCL
   );
```

Параметры

program

[in] Хендл объекта OpenCL.

Возвращаемое значение

Нет. В случае внутренней ошибки изменяется значение _LastError. Для получения информации об ошибке используйте функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLKernelCreate

Создает точку входа в программу OpenCL и возвращает хендл на неё.

```
int  CLKernelCreate(
   int           program,        // хендл на объект OpenCL
   const string  kernel_name     // имя кернела
   );
```

Параметры

program

[in] Хендл на объект программы OpenCL.

kernel_name

[in] Имя функции кернел, то есть имя точки входа в соответствующей OpenCL-программе, с которой начинается выполнение.

Возвращаемое значение

Хендл на объект OpenCL при успешном выполнении. В случае ошибки возвращает -1. Для получения информации об ошибке используйте функцию GetLastError().

Примечание

На данный момент предусмотрены следующие коды ошибок:

- ERR_OPENCL_INVALID_HANDLE - невалидный хендл OpenCL-программы,
- ERR_INVALID_PARAMETER - невалидный строковой параметр,
- ERR_OPENCL_TOO_LONG_KERNEL_NAME - имя кернела содержит более 127 символов,
- ERR_OPENCL_KERNEL_CREATE - внутренняя ошибка при создании объекта OpenCL.

© 2000-2025, MetaQuotes Ltd.

CLKernelFree

Удаляет функцию запуска OpenCL.

```
void  CLKernelFree(
   int  kernel     // хендл на кернел OpenCL программы
   );
```

Параметры

kernel_name

[in] Хендл объекта кернел.

Возвращаемое значение

Нет. В случае внутренней ошибки изменяется значение _LastError. Для получения информации об ошибке используйте функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLSetKernelArg

Выставляет параметр для функции OpenCL.

```
bool  CLSetKernelArg(
   int   kernel,         // хендл на кернел OpenCL программы
   uint  arg_index,      // номер аргумента OpenCL функции
   void  arg_value       // исходный код
   );
```

Параметры

kernel

[in] Хендл на кернел программы OpenCL.

arg_index

[in] Номер аргумента функции, нумерация начинается с нуля.

arg_value

[in] Значение аргумента функции.

Возвращаемое значение

Возвращает true при успешном выполнении, в противном случае false. Для получения информации об ошибке используйте функцию GetLastError().

Примечание

На данный момент предусмотрены следующие коды ошибок:

- ERR_INVALID_PARAMETER,
- ERR_OPENCL_INVALID_HANDLE - невалидный хендл на кернел OpenCL,
- ERR_OPENCL_SET_KERNEL_PARAMETER - внутренняя ошибка OpenCL.

© 2000-2025, MetaQuotes Ltd.

CLSetKernelArgMem

Выставляет буфер OpenCL в качестве параметра функции OpenCL.

```
bool  CLSetKernelArgMem(
   int   kernel,           // хендл на кернел OpenCL программы
   uint  arg_index,        // номер аргумента OpenCL функции
   int   cl_mem_handle     // хендл буфера OpenCL
   );
```

Параметры

kernel

[in] Хендл на кернел программы OpenCL.

arg_index

[in] Номер аргумента функции, нумерация начинается с нуля.

cl_mem_handle

[in] Хендл на буфер OpenCL.

Возвращаемое значение

Возвращает true при успешном выполнении, в противном случае false. Для получения информации об ошибке используйте функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLSetKernelArgMemLocal

Задаёт локальный буфер в качестве аргумента kernel-функции.

```
bool  CLSetKernelArgMemLocal(
   int    kernel,           // хендл на кернел OpenCL программы
   uint   arg_index,        // номер аргумента OpenCL функции
   ulong  local_mem_size    // размер буфера
   );
```

Параметры

kernel

[in] Хендл на кернел программы OpenCL.

arg_index

[in] Номер аргумента функции, нумерация начинается с нуля.

local_mem_size

[in] Размер буфера в байтах.

Возвращаемое значение

Возвращает true при успешном выполнении, в противном случае false. Для получения информации об ошибке используйте функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLBufferCreate

Создает буфер OpenCL и возвращает хендл на него.

```
int  CLBufferCreate(
   int   context,     // хендл на контекст OpenCL
   uint  size,        // размер буфера
   uint  flags        // комбинация флагов, задающие свойства буфера
   );
```

Параметры

context

[in] Хендл на context OpenCL.

size

[in] Размер буфера в байтах.

flags

[in] Свойства буфера, задаваемые через комбинацию флагов: CL_MEM_READ_WRITE, CL_MEM_WRITE_ONLY, CL_MEM_READ_ONLY, CL_MEM_ALLOC_HOST_PTR.

Возвращаемое значение

Хендл на буфер OpenCL при успешном выполнении. В случае ошибки возвращает -1. Для получения информации об ошибке используйте функцию GetLastError().

Примечание

На данный момент предусмотрены следующие коды ошибок:

- ERR_OPENCL_INVALID_HANDLE - невалидный хендл на контекст OpenCL,
- ERR_NOT_ENOUGH_MEMORY - недостаточно памяти,
- ERR_OPENCL_BUFFER_CREATE - внутренняя ошибка создания буфера.

© 2000-2025, MetaQuotes Ltd.

CLBufferFree

Удаляет буфер OpenCL.

```
void  CLBufferFree(
   int   buffer     // хендл на буфер OpenCL
   );
```

Параметры

buffer

[in] Хендл на буфер OpenCL.

Возвращаемое значение

Нет. В случае внутренней ошибки изменяется значение _LastError. Для получения информации об ошибке используйте функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

CLBufferWrite

Записывает массив в буфер OpenCL и возвращает количество записанных элементов.

```
uint  CLBufferWrite(
   int           buffer,                    // хендл на буфер OpenCL
   const void&   data[],                    // массив значений
   uint          buffer_offset=0,           // смещение в OpenCL буфере в байтах, по ум
   uint          data_offset=0,             // смещение в массиве в элементах, по умолч
   uint          data_count=WHOLE_ARRAY     // количество значений из массива для запис
   );
```

Существуют также версии для работы матрицами и векторами.

Записывает значения из матрицы в буфер и возвращает true в случае успеха.

```
uint  CLBufferWrite(
   int           buffer,                    // хендл на буфер OpenCL
   uint          buffer_offset,             // смещение в OpenCL буфере в байтах
   matrix<T>     &mat                       // матрица значений для записи в буфер
   );
```

Записывает значения из вектора в буфер и возвращает true в случае успеха.

```
uint  CLBufferWrite(
   int           buffer,                    // хендл на буфер OpenCL
   uint          buffer_offset,             // смещение в OpenCL буфере в байтах
   vector<T>     &vec                       // вектор значений для записи в буфер
   );
```

Параметры

buffer

[in] Хендл буфера OpenCL.

data[]

[in] Массив значений, которые необходимо записать в буфер OpenCL. Передается по ссылке.

buffer_offset

[in] Смещение в OpenCL буфере в байтах, с которого начинается запись. По умолчанию запись идет с самого начала буфера.

data_offset

[in] Индекс первого элемента массива, начиная с которого берутся значения из массива для записи в OpenCL буфер. По умолчанию значения берутся с самого начала массива.

data_count

[in] Количество значений, которые нужно записать. По умолчанию все значения массива.

mat

[out] Матрица для чтения данных из буфера может быть любого из трех типиов — matrix, matrixf или matrixc.

© 2000-2025, MetaQuotes Ltd.

vec

[out] Вектор для чтения данных из буфера может быть любого из трех типиов — vector, vectorf или vectorc.

Возвращаемое значение

Количество записанных элементов, в случае ошибки возвращается 0. Для получения информации об ошибке используйте функцию GetLastError().

true при успешном выполнении при работе с матрицей или вектором, в случае ошибки возвращается false.

Примечание

Для одномерных массивов номер элемента, с которого начинается чтение данных для записи в буфер OpenCL, вычисляется с учётом флага AS_SERIES.

Массив с размерностью два и более представляется как одномерный. В этом случае *data_offset* - это количество элементов, которое следует пропустить в представлении, а не количество элементов в первой размерности.

Пример умножения матриц с помощью метода MatMul и параллельных вычислений в OpenCL

```
#define M       3000      // число строк в первой матрице
#define K       2000      // число столбцов в первой матрице равно числу строк во втор
#define N       3000      // число столбцов во второй матрице

//+------------------------------------------------------------------+
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()
 {
//--- инициализируем генератор случайных чисел
  MathSrand((int)TimeCurrent());
```

© 2000-2025, MetaQuotes Ltd.

```
//--- заполним матрицы заданного размера случайными значениями
  matrixf mat1(M, K, MatrixRandom) ;     // первая матрица
  matrixf mat2(K, N, MatrixRandom);      // вторая матрица

//--- посчитаем произведение матриц наивным способом
  uint start=GetTickCount();
  matrixf matrix_naive=matrixf::Zeros(M, N);// сюда запишем результат умножения двух м
  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;

//--- посчитаем произведение матриц через MatMull
  start=GetTickCount();
  matrixf matrix_matmul=mat1.MatMul(mat2);
  uint time_matmul=GetTickCount()-start;

//--- посчитаем произведение матриц в OpenCL
  matrixf matrix_opencl=matrixf::Zeros(M, N);
  int cl_ctx;              // хэндл контекста
  if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL не найдено, выходим");
    return;
   }
  int cl_prg;              // хэндл программы
  int cl_krn;              // хэндл кернела
  int cl_mem_in1;          // хэндл первого буфера (входного)
  int cl_mem_in2;          // хэндл второго буфера (входного)
  int cl_mem_out;          // хэндл третьего буфера (выходного)
//--- создаем программу и кернел
  cl_prg = CLProgramCreate(cl_ctx, clSrc);
  cl_krn = CLKernelCreate(cl_prg, "matricesMul");
//--- создаем все три буфера для трех матриц
  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);
//--- третья матрица - выходная
  cl_mem_out=CLBufferCreate(cl_ctx, M*N*sizeof(float), CL_MEM_READ_WRITE);
//--- устанавливаем аргументы кернела
  CLSetKernelArgMem(cl_krn, 0, cl_mem_in1);
  CLSetKernelArgMem(cl_krn, 1, cl_mem_in2);
  CLSetKernelArgMem(cl_krn, 2, cl_mem_out);
//--- пишем матрицы в буферы девайса
  CLBufferWrite(cl_mem_in1, 0, mat1);
  CLBufferWrite(cl_mem_in2, 0, mat2);
  CLBufferWrite(cl_mem_out, 0, matrix_opencl);
//--- старт времени исполнения кода OpenCL
  start=GetTickCount();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- устанавливаем параметры рабочего пространства задачи и исполняем программу OpenC
  uint  offs[2] = {0, 0};
  uint works[2] = {M, N};
  start=GetTickCount();
  bool ex=CLExecute(cl_krn, 2, offs, works);
//--- считываем результат в матрицу
  if(CLBufferRead(cl_mem_out, 0, matrix_opencl))
    PrintFormat("Прочитана матрица [%d x %d]: ", matrix_opencl.Rows(), matrix_opencl.C
   else
      Print("CLBufferRead(cl_mem_out, 0, matrix_opencl failed. Error ",GetLastError())
  uint time_opencl=GetTickCount()-start;
  Print("Сравним время вычислений каждым методом");
  PrintFormat("Naive product time = %d ms",time_naive);
  PrintFormat("MatMul product time = %d ms",time_matmul);
  PrintFormat("OpenCl product time = %d ms",time_opencl);
//--- освободим все OpenCL контексты
  CLFreeAll(cl_ctx, cl_prg, cl_krn, cl_mem_in1, cl_mem_in2, cl_mem_out);

//--- сравним все полученные матрицы результатов между собой
  Print("Сколько ошибок расхождения между матрицами результатов?");
  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);
/*
   Результат:

   Прочитана матрица [3000 x 3000]:
   Сравним время вычислений каждым методом
   Naive product time = 54750 ms
   MatMul product time = 4578 ms
   OpenCl product time = 922 ms
   Сколько ошибок расхождения между матрицами результатов?
   matrix_direct.Compare(matrix_matmul,1e-12)=0
   matrix_matmul.Compare(matrix_opencl,1e-12)=0
*/
 }
//+------------------------------------------------------------------+
//| Заполняет матрицу случайными значениями                          |
//+------------------------------------------------------------------+
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.);
     }
   }
```

© 2000-2025, MetaQuotes Ltd.

```
 }
//+------------------------------------------------------------------+
//| Освободим все OpenCL контексты                                   |
//+------------------------------------------------------------------+
void CLFreeAll(int cl_ctx, int cl_prg, int cl_krn,
               int cl_mem_in1, int cl_mem_in2, int cl_mem_out)
 {
//--- удвляем в обратной последовательности все созданные OpenCL контексты
  CLBufferFree(cl_mem_in1);
  CLBufferFree(cl_mem_in2);
  CLBufferFree(cl_mem_out);
  CLKernelFree(cl_krn);
  CLProgramFree(cl_prg);
  CLContextFree(cl_ctx);
 }
```

© 2000-2025, MetaQuotes Ltd.

CLBufferRead

Читает буфер OpenCL в массив и возвращает количество прочитанных элементов.

```
uint  CLBufferRead(
   int           buffer,                    // хендл на буфер OpenCL
   const void&   data[],                    // массив значений
   uint          buffer_offset=0,           // смещение в OpenCL буфере в байтах, по ум
   uint          data_offset=0,             // смещение в массиве в элементах, по умолч
   uint          data_count=WHOLE_ARRAY     // количество значений из буфера для чтения
   );
```

Существуют также версии для работы матрицами и векторами.

Читает буфер OpenCL в матрицу и возвращает true в случае успеха.

```
uint  CLBufferRead(
   int           buffer,                    // хендл на буфер OpenCL
   uint          buffer_offset,             // смещение в OpenCL буфере в байтах
   const matrix& mat,                       // матрица для получения значений  из буфе
   ulong         rows=-1,                   // количество строк в матрице
   ulong         cols=-1                    // количество столбцов в матрице
   );
```

Читает буфер OpenCL в вектор и возвращает true в случае успеха.

```
uint  CLBufferRead(
   int           buffer,                    // хендл на буфер OpenCL
   uint          buffer_offset,             // смещение в OpenCL буфере в байтах
   const vector& vec,                       // вектор для получения значений из буфера
   ulong         size-1,                    // длина вектора
   );
```

Параметры

buffer

[in] Хендл буфера OpenCL.

data[]

[out] Массив для получения значений из буфера OpenCL. Передается по ссылке.

buffer_offset

[in] Смещение в OpenCL буфере в байтах, с которого начинается чтение. По умолчанию чтение начинается с начала буфера.

data_offset

[in] Индекс первого элемента массива для записи значений буфера OpenCL. По умолчанию запись прочитанных значений в массив начинается с нулевого индекса.

data_count

[in] Количество значений, которые нужно прочитать. По умолчанию читается весь буфер OpenCL.

© 2000-2025, MetaQuotes Ltd.

mat

[out] Матрица для чтения данных из буфера может быть любого из трех типов — matrix, matrixf или matrixc.

vec

[out] Вектор для чтения данных из буфера может быть любого из трех типов — vector, vectorf или vectorc.

rows=-1

[in] Если параметр указан, то должен быть указан и параметр *cols*. Если новые размеры матрицы не указаны, то будут использоваться текущие размеры. Если значение равно -1, то число строк не изменится.

cols=-1

[in] Если параметр не указан, то параметр *rows* также должен быть пропущен. Для матрицы действует правило: либо указаны оба параметра, либо ни одного, в противном случае возникнет ошибка. Если указаны об параметра (*rows и cols*), то размер матрицы будет изменен. Если значение равно -1, то число столбцов не изменится.

size=-1

[in] Если параметр не указан или значение равно -1, то длина вектора не изменится.

Возвращаемое значение

Количество прочитанных элементов при работе с массивом, в случае ошибки возвращается 0. Для получения информации об ошибке используйте функцию GetLastError().

true при успешном выполнении при работе с матрицей или вектором, в случае ошибки возвращается false.

Примечание

Для одномерных массивов номер элемента, в который начинается запись данных из буфера OpenCL, вычисляется с учётом флага AS_SERIES.

Массив с размерностью два и более представляется как одномерный. В этом случае *data_offset* - это количество элементов, которое следует пропустить в представлении, а не количество элементов в первой размерности.

Пример вычисления числа Pi по формуле:

$$\pi = \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N-1}\frac{4}{1+\left(\frac{2k+1}{2N}\right)^2} = 16\lim_{N\to\infty} N\sum_{k=0}^{N-1}\frac{1}{4N^2+(2k+1)^2}$$

```
#define  _num_steps        1000000000
#define  _divisor          40000
#define  _step             1.0 / _num_steps
```

© 2000-2025, MetaQuotes Ltd.

```
#define  _intrnCnt          _num_steps / _divisor

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
string D2S(double arg, int digits) {return DoubleToString(arg, digits);}
string I2S(int arg)                {return IntegerToString(arg);}

//--- код OpenCL программы
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));
//--- создадим OpenCL контексты
  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};
//--- запускаем OpenCL программу
  ulong start=GetMicrosecondCount();
```

© 2000-2025, MetaQuotes Ltd.

```
  if(!CLExecute(clKrn, 1, offs, works))
   {
    Print("CLExecute(clKrn, 1, offs, works) failed! Error ", GetLastError());
    CLFreeAll(clMem, clKrn, clPrg, clCtx);
    return(-1);
   }
//--- получим результаты вычисления из OpenCL
  vector buffer(_divisor);
  if(!CLBufferRead(clMem, 0, buffer))
   {
    Print("CLBufferRead(clMem, 0, buffer) failed! Error ", GetLastError());
    CLFreeAll(clMem, clKrn, clPrg, clCtx);
    return(-1);
   }
//--- просуммируем все значения для вычисления 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));
//--- освободим все OpenCL контексты
  CLFreeAll(clMem, clKrn, clPrg, clCtx);
//--- успешно
  return(0);
 }
  /*
  Pi Calculation: step = 0.000000001000; _intrnCnt = 25000
  OpenCL: GPU device 'Ellesmere' selected
  OpenCL: Pi calculated for 99.98 ms
  Pi = 3.141592653590
  */
//+------------------------------------------------------------------+
//| Освободим все OpenCL контексты                                   |
//+------------------------------------------------------------------+
void CLFreeAll(const int clMem, const int clKrn, const int clPrg, const int clCtx)
 {
  CLBufferFree(clMem);
  CLKernelFree(clKrn);
  CLProgramFree(clPrg);
  CLContextFree(clCtx);
 }
```

© 2000-2025, MetaQuotes Ltd.

CLExecute

Выполняет OpenCL программу. Существует 3 варианта функции:

1. Запуск функции kernel на одном ядре

```
bool  CLExecute(
   int           kernel                    // хендл на кернел OpenCL программы
   );
```

2. Запуск нескольких копий kernel (OpenCL функция) с описанием пространства задач

```
bool  CLExecute(
   int           kernel,                   // хендл на кернел OpenCL программы
   uint          work_dim,                 // размерность пространства задач
   const uint&   global_work_offset[],     // начальное смещение в пространстве задач
   const uint&   global_work_size[]        // общее количество задач
   );
```

3. Запуск нескольких копий kernel (OpenCL функция) с описанием пространства задач и указанием размера локального подмножества задач в группе

```
bool  CLExecute(
   int           kernel,                   // хендл на кернел OpenCL программы
   uint          work_dim,                 // размерность пространства задач
   const uint&   global_work_offset[],     // начальное смещение в пространстве задач
   const uint&   global_work_size[],       // общее количество задач
   const uint&   local_work_size[]         // количество задач в локальной группе
   );
```

Параметры

kernel

[in] Хендл на кернел OpenCL.

work_dim

[in] Размерность пространства задач.

global_work_offset[]

[in] Начальное смещение в пространстве задач.

global_work_size[]

[in] Размер подмножества задач.

local_work_size[]

[in] Размер локального подмножества задач в группе.

Возвращаемое значение

Возвращает true при успешном выполнении, в противном случае false. Для получения информации об ошибке используйте функцию GetLastError().

Примечание

© 2000-2025, MetaQuotes Ltd.

Рассмотрим на примере смысл параметров:

- *work_dim* задает размерность массива *work_items[]*, описывающего задачи. Если *work_dim*=3, то используется 3-мерный массив *work_items[N1, N2, N3]*.
- *global_work_size[]* содержит значения, задающие размер массива *work_items[]*. Если у нас work_dim=3, и соответственно, массив *global_work_size[3]* может быть {40, 100, 320}. Тогда имеем *work_items[40, 100, 320]*. Значит общее количество задач равно 40 x 100 x 320 = 1 280 000.
- *local_work_size[]* задает подмножество задач, которые будут выполняться указанным кернелом OpenCL программы. Его размерность равна размерности *work_items[]* и позволяет общее подмножество задач нарезать на более мелкие подмножества без остатков от деления. Фактически, размеры массива *local_work_size[]* должны быть подобраны таким образом, чтобы глобальное множество задач *work_items[]* нарезалось на более мелкие подмножества. В данном примере подойдет *local_work_size[3]={10, 10, 10}*, так как *work_items[40, 100, 320]* без остатка можно собрать из массива *local_items[10, 10, 10]*.

© 2000-2025, MetaQuotes Ltd.

CLExecutionStatus

Возвращает состояние выполнения OpenCL программы.

```
int  CLExecutionStatus(
   int   kernel            // хендл на кернел OpenCL программы
   );
```

Параметры

kernel

[in] Хендл на кернел программы OpenCL.

Возвращаемое значение

Возвращает статус OpenCL программы, значение может быть одним из нижеследующих:

- CL_COMPLETE=0 - программа завершена,
- CL_RUNNING=1 - выполняется,
- CL_SUBMITTED=2 - отправлена на выполнение,
- CL_QUEUED=3 - находится в очереди на выполнение,
- -1 (минус один) - произошла ошибка при выполнении CLExecutionStatus().

© 2000-2025, MetaQuotes Ltd.

Работа с базами данных

Функции для работы с базами данных используют популярный и простой в работе движок SQLite. Удобством данного движка является то, что вся база данных находится в единственном файле на жестком диске на компьютере пользователя.

С помощью этих функций легко создавать таблицы, добавлять в них данные, производить модификации и делать выборки простыми SQL-запросами:

- получение торговой истории и котировок из любых форматов,
- сохранение результатов оптимизации и тестирования,
- подготовка и обмен данными с другими пакетами анализа,
- хранение настроек и состояния MQL5-программ.

При этом в запросах можно использовать статистические и математические функции.

Функции для работы с базами позволяют заменить наиболее повторяющиеся операции по обработке больших массивов данных на SQL-запросы, что позволит во многих случаях вместо программирования сложных циклов и сравнений использовать вызовы DatabaseExecute/DatabasePrepare. Для удобного получения результатов запроса в готовую структуру используйте функцию DatabaseReadBind, которая позволит прочитать сразу все поля записи за один вызов.

Для ускорения операций чтения, записи и модификации базу данных можно открывать/создавать в оперативной памяти с флагом DATABASE_OPEN_MEMORY, но при этом такая база будет доступна только данной конкретной программе и ни с кем не разделяется. При работе с базами данных, которые находятся на жестком диске, необходимо обертывать массовые вставки/изменения данных в транзакции с помощью DatabaseTransactionBegin/DatabaseTransactionCommit/DatabaseTransactionRollback, что дает ускорение в сотни раз.

Для того чтобы начать работать с данными функциями, достаточно прочитать статью SQLite: нативная работа с базами данных на SQL в MQL5.

Функция	Действие
DatabaseOpen	Открывает или создаёт базу данных в указанном файле
DatabaseClose	Закрывает базу данных
DatabaseImport	Импортирует в таблицу данные из файла
DatabaseExport	Экспортирует таблицу или результат выполнения SQL-запроса в CSV-файл
DatabasePrint	Печатает таблицу или результат выполнения SQL-запроса в журнал экспертов
DatabaseTableExists	Проверяет наличие таблицы в базе данных
DatabaseExecute	Исполняет запрос к указанной базе данных
DatabasePrepare	Создает хендл запроса, который затем может быть исполнен с помощью DatabaseRead()

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
DatabaseReset	Сбрасывает запрос в начальное состояние, как после вызова DatabasePrepare()
DatabaseBind	Устанавливает значение параметра в запросе
DatabaseBindArray	Устанавливает массив в качестве значения параметра
DatabaseRead	Выполняет переход к следующей записи в результате запроса
DatabaseReadBind	Выполняет переход к следующей записи и читает из неё данные в структуру
DatabaseFinalize	Удаляет запрос, созданный в DatabasePrepare()
DatabaseTransactionBegin	Начинает выполнение транзакции
DatabaseTransactionCommit	Завершает выполнение транзакции
DatabaseTransactionRollback	Выполняет откат транзакций
DatabaseColumnsCount	Получает количество полей в запросе
DatabaseColumnName	Получает имя поля по номеру
DatabaseColumnType	Получает тип поля по номеру
DatabaseColumnSize	Получает размер поля в байтах
DatabaseColumnText	Получает из текущей записи значение поля в виде строки
DatabaseColumnInteger	Получает из текущей записи значение типа int
DatabaseColumnLong	Получает из текущей записи значение типа long
DatabaseColumnDouble	Получает из текущей записи значение типа double
DatabaseColumnBlob	Получает из текущей записи значение поля в виде массива

Статистические функции:

- mode — мода
- median — медиана (50-й перцентиль)
- percentile_25 — 25-й перцентиль
- percentile_75 — 75-й перцентиль
- percentile_90 — 90-й перцентиль
- percentile_95 — 95-й перцентиль
- percentile_99 — 99-й перцентиль
- stddev или stddev_samp — стандартное отклонение выборки
- stddev_pop — стандартное отклонение популяции
- variance or var_samp — дисперсия выборки
- var_pop — дисперсия популяции

© 2000-2025, MetaQuotes Ltd.

Математические функции

- acos(X) - арккосинус в радианах
- acosh(X) - арккосинус гиперболический
- asin(X) - арксинус в радианах
- asinh(X) - арксинус гиперболический
- atan(X) - арктангенс в радианах
- atan2(X,Y) - арктангенс в радианах отношения X/Y
- atanh(X) - арктангенс гиперболический
- ceil(X) - округление до целого числа вверх
- ceiling(X) - округление до целого числа вверх
- cos(X) - косинус угла в радианах
- cosh(X) - косинус гиперболический
- degrees(X) - конвертирует радианы в угол
- exp(X) - экспонента
- floor(X) - округление до целого числа вниз
- ln(X) - натуральный логарифм
- log(B,X) - логарифм по указанному основанию
- log(X) - десятичный логарифм
- log10(X) - десятичный логарифм
- log2(X) - логарифм по основанию 2
- mod(X,Y) - остаток от деления
- pi() - приближенное число Pi
- pow(X,Y) - степень по заданному основанию
- power(X,Y) - степень по заданному основанию
- radians(X) - конвертирует угол в радианы
- sin(X) - синус угла в радианах
- sinh(X) - синус гиперболический
- sqrt(X) - квадратный корень
- tan(X) - тангенс угла в радианах
- tanh(X) - тангенс гиперболический
- trunc(X) - усекает до целого числа, наиболее близкого к 0

Пример:

```
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,
```

© 2000-2025, MetaQuotes Ltd.

```
  percentile_99(parent) as p99
from moz_bookmarks;
```

© 2000-2025, MetaQuotes Ltd.

DatabaseOpen

Открывает или создаёт базу данных в указанном файле.

```
int  DatabaseOpen(
   string  filename,       // имя файла
   uint    flags           // комбинация флагов
   );
```

Параметры

filename

[in] Имя файла относительно папки "MQL5\Files".

flags

[in] Комбинация флагов из перечисления ENUM_DATABASE_OPEN_FLAGS.

Возвращаемое значение

При успешном выполнении функция возвращает хэндл базы данных, который затем используется для доступа к данным базы, в противном случае возвращает INVALID_HANDLE. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INTERNAL_ERROR (4001) - критическая ошибка исполняющей системы;
- ERR_WRONG_INTERNAL_PARAMETER (4002) - внутренняя ошибка доступа к папке "MQL5\Files";
- ERR_INVALID_PARAMETER (4003) - путь к файлу базы содержит пустую строку или выставлена несовместимая комбинация флагов;
- ERR_NOT_ENOUGH_MEMORY (4004) - недостаточно памяти;
- ERR_WRONG_FILENAME (5002) - некорректное имя файла базы данных;
- ERR_TOO_LONG_FILENAME (5003) - абсолютный путь к файлу базы данных превысил максимальную длину;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - превышено максимально допустимое количество объектов Database;
- ERR_DATABASE_CONNECT (5123) - ошибка подключения к базе данных;
- ERR_DATABASE_MISUSE (5621) - неправильное использование библиотеки SQLite.

Примечание

Если в параметре *filename* указан NULL или пустая строка "", то на диске будет создан временный файл, который будет автоматически удален после закрытия соединения с базой данных.

Если в параметре *filename* указан ":memory:", то база данных будет создана в памяти, при этом база данных будет автоматически удалена после закрытия соединения с ней.

Если в параметре *flags* не указан ни один из флагов DATABASE_OPEN_READONLY или DATABASE_OPEN_READWRITE, то будет использован флаг DATABASE_OPEN_READWRITE.

Если для файла не задано расширение, то будет использовано расширение ".sqlite"

ENUM_DATABASE_OPEN_FLAGS

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
DATABASE_OPEN_READONLY	Открыть только на чтение
DATABASE_OPEN_READWRITE	Открыть на чтение и запись
DATABASE_OPEN_CREATE	Создать файл на диске, если он не существует
DATABASE_OPEN_MEMORY	Создать базу данных в оперативной памяти
DATABASE_OPEN_COMMON	Файл находится в общей папке всех терминалов

Смотри также

DatabaseClose

© 2000-2025, MetaQuotes Ltd.

DatabaseClose

Закрывает базу данных.

```
void  DatabaseClose(
   int  database      // хендл базы данных, полученный в DatabaseOpen
   );
```

Параметры

database

[in] Хендл базы данных, полученный в DatabaseOpen().

Возвращаемое значение

Нет.

Примечание

После вызова DatabaseClose все хендлы запросов к базе удаляются автоматически и становятся невалидными.

В случае невалидного хендла функция выставит ошибку ERR_DATABASE_INVALID_HANDLE. Проверить ошибку можно с помощью GetLastError().

Смотри также

DatabaseOpen, DatabasePrepare

© 2000-2025, MetaQuotes Ltd.

DatabaseImport

Импортирует в таблицу данные из файла.

```
long  DatabaseImport(
   int           database,          // хендл базы данных, полученный в DatabaseOpen
   const string  table,             // имя таблицы для вставки данных
   const string  filename,          // имя файла для импорта данных
   uint          flags,             // комбинация флагов
   const string  separator,         // разделитель данных
   ulong         skip_rows,         // сколько первых строк пропустить
   const string  skip_comments      // строка символов, которые определяют комментарии
   );
```

Параметры

database

[in] Хендл базы данных, который получен в DatabaseOpen().

table

[in] Имя таблицы, в которую будут добавлены данные из файла.

filename

[in] CSV-файл или ZIP-архив для чтения данных, имя может содержать подпапки и задается относительно папки MQL5\Files.

flags

[in] Комбинация флагов из перечисления ENUM_DATABASE_IMPORT_FLAGS.

separator

[in] Разделитель данных в CSV-файле.

skip_rows

[in] Количество начальных строк, которые необходимо пропустить при чтении данных из файла.

skip_comments

[in] Строка из символов для обозначения строк как комментариев. Если в начале строки найден любой символ из *skip_comments*, то такая строка считается комментарием и не импортируется.

Возвращаемое значение

Возвращает количество импортированных строк или -1 в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INVALID_PARAMETER (4003) - имя таблицы не задано (пустая строка или NULL);
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных.

Примечание

© 2000-2025, MetaQuotes Ltd.

Если таблицы с именем *table* не существует, то она будет создана автоматически. Имена и тип полей в созданной таблице будут распознаны автоматически на основе данных, содержащихся в файле.

ENUM_DATABASE_IMPORT_FLAGS

Идентификатор	Описание
DATABASE_IMPORT_HEADER	Первая строка содержит имена полей таблицы
DATABASE_IMPORT_CRLF	Переносом строки считается CRLF (по умолчанию LF)
DATABASE_IMPORT_APPEND	Добавлять данные в конец существующей таблицы
DATABASE_IMPORT_QUOTED_STRINGS	Строковые значения заключены в двойных кавычках
DATABASE_IMPORT_COMMON_FOLDER	Файл находится в общей папке всех клиентских терминалов \Terminal\Common\File.

Пример чтения таблицы из файла, созданного кодом из примера в DatabaseExport :

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  string csv_filename;
//--- получим имена текстовых файлов для загрузки из общей папки клиентских терминалов
  string filenames[];
  if(FileSelectDialog("Выберите CSV-файл для загрузки таблицы", NULL,
                      "Text files (*.csv)|*.csv",
                      FSD_WRITE_FILE|FSD_COMMON_FOLDER, filenames, "data.csv")>0)
   {
    //--- выведем имя каждого выбранного файла
    if(ArraySize(filenames)==1)
      csv_filename=filenames[0];
    else
     {
      Print("Неизвестная ошибка при выборе файла. Код ошибки ", GetLastError());
      return;
     }
   }
  else
   {
    Print("CSV-файл не выбран");
    return;
   }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- создадим или откроем базу данных
  string db_filename="test.sqlite";
  int db=DatabaseOpen(db_filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE);
//--- проверим наличие таблицы TEST
  if(DatabaseTableExists(db, "TEST"))
    {
     //--- удалим таблицу TEST
     if(!DatabaseExecute(db, "DROP TABLE IF EXISTS TEST"))
      {
       Print("Failed to drop the TEST table with code ", GetLastError());
       DatabaseClose(db);
       return;
      }
    }
  //--- импортируем записи из файла в таблицу 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());
     }
//--- закроем файл с базой данных и сообщим об этом
  DatabaseClose(db);
  PrintFormat("Database: %s closed", db_filename);
 }
```

Смотри также

DatabaseOpen, DatabasePrint

© 2000-2025, MetaQuotes Ltd.

DatabaseExport

Экспортирует таблицу или результат выполнения SQL-запроса в CSV-файл. Файл создается в кодировке UTF-8.

```
long  DatabaseExport(
   int           database,          // хендл базы данных, полученный в DatabaseOpen
   const string  table_or_sql,      // имя таблицы или SQL-запрос
   const string  filename,          // имя CSV-файла для экспорта данных
   uint          flags,             // комбинация флагов
   const string  separator          // разделитель данных в CSV-файле
   );
```

Параметры

database

[in] Хендл базы данных, который получен в DatabaseOpen().

table_or_sql

[in] Имя таблицы или текст SQL-запроса, результаты которого будут экспортированы в указанный файл.

filename

[in] Имя файла для экспорта данных. Путь задается относительно папки MQL5\Files.

flags

[in] Комбинация флагов из перечисления ENUM_DATABASE_EXPORT_FLAGS.

separator

[in] Разделитель данных. Если указан NULL, то в качестве разделителя будет использоваться символ табуляции '\t'. Пустая строка "" считается допустимым разделителем, но полученный CSV-файл не может быть прочитан как таблица - это будет набор строк.

Возвращаемое значение

Возвращает количество экспортированных записей или отрицательное значение в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INTERNAL_ERROR (4001) - критическая ошибка исполняющей системы;
- ERR_INVALID_PARAMETER (4003) - путь к файлу базы содержит пустую строку или выставлена несовместимая комбинация флагов;
- ERR_NOT_ENOUGH_MEMORY (4004) - недостаточно памяти;
- ERR_FUNCTION_NOT_ALLOWED(4014) - указанный пайп не разрешён;
- ERR_PROGRAM_STOPPED(4022) - операция отменена (произошла остановка MQL программы);
- ERR_WRONG_FILENAME (5002) - некорректное имя файла;
- ERR_TOO_LONG_FILENAME (5003) - абсолютный путь к файлу превысил максимальную длину;
- ERR_CANNOT_OPEN_FILE(5004) - ошибка открытия файла на запись;
- ERR_FILE_WRITEERROR(5026) - ошибка записи в файл;
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;

© 2000-2025, MetaQuotes Ltd.

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_QUERY_PREPARE(5125) - ошибка создания запроса;
- ERR_DATABASE_QUERY_NOT_READONLY - разрешен только запрос на чтение.

Примечание

Если экспортируются результаты запроса, то SQL-запрос должен начинаться с "SELECT" или "select". Другими словами, SQL-запрос не может изменять состояние базы данных, в противном случае DatabaseExport() завершится ошибкой.

Строковые значения в базе данных могут содержать символ перевода ('\r' или '\r\n'), а также символ разделителя значений, заданный в параметре *separator*. В этом случае нужно обязательно использовать флаг DATABASE_EXPORT_QUOTED_STRINGS в параметре flags. При наличии этого флага все выводимые строки будут заключены в двойные кавычки, если же в строке содержится двойная кавычка, то она будет заменена на две двойные кавычки.

ENUM_DATABASE_EXPORT_FLAGS

Идентификатор	Описание
DATABASE_EXPORT_HEADER	Выводить имена полей в первой строке
DATABASE_EXPORT_INDEX	Выводить номера строк
DATABASE_EXPORT_NO_BOM	Не вставлять метку BOM в начале файла (по умолчанию BOM вставляется)
DATABASE_EXPORT_CRLF	Для переноса строки использовать CRLF (по умолчанию LF)
DATABASE_EXPORT_APPEND	Дописывать данные в конец существующего файла (по умолчанию файл перезаписывается). Если файл не существует, то он будет создан.
DATABASE_EXPORT_QUOTED_STRINGS	Выводить строковые значения в двойных кавычках.
DATABASE_EXPORT_COMMON_FOLDER	CSV-файл будет создан в общей папке всех клиентских терминалов \Terminal\Common\File.

Пример:

```
input int InpRates=100;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  MqlRates rates[];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- запомним время старта перед получением баров
  ulong start=GetMicrosecondCount();
//--- запросим последние 100 баров на тайфрейме H1
  if(CopyRates(Symbol(), PERIOD_H1, 1, InpRates, rates)<InpRates)
    {
     Print("CopyRates() failed,, Error ", GetLastError());
     return;
    }
  else
    {
     //--- сколько баров и за какое время получили
     PrintFormat("%s: CopyRates received %d bars in %d ms ",
                 _Symbol, ArraySize(rates), (GetMicrosecondCount()-start)/1000);
    }
//--- составим имя файла для хранения базы данных
  string filename=_Symbol+"_"+EnumToString(PERIOD_H1)+"_"+TimeToString(TimeCurrent())+
  StringReplace(filename, ":", "-"); // символ ":" запрещен в названиях файлов
//--- открываем/создаем базу данных в общей папке терминалов
  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");

//--- проверим наличие таблицы RATES
  if(DatabaseTableExists(db, "RATES"))
    {
     //--- удалим таблицу RATES
     if(!DatabaseExecute(db, "DROP TABLE IF EXISTS RATES"))
       {
        Print("Failed to drop the RATES table with code ", GetLastError());
        DatabaseClose(db);
        return;
       }
    }
//--- создаем таблицу 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);"))
```

© 2000-2025, MetaQuotes Ltd.

```
    {
     Print("DB: ", filename, " create table RATES with code ", GetLastError());
     DatabaseClose(db);
     return;
    }
//--- покажем список всех полей в таблице RATES
  if(DatabasePrint(db, "PRAGMA TABLE_INFO(RATES)", 0)<0)
    {
     PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(RATES)\") failed, error code=%d at
     DatabaseClose(db);
     return;
    }
//--- создадим параметризованный запрос добавления баров в таблицу 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)"; // параметры запроса
  int request=DatabasePrepare(db, sql);
  if(request==INVALID_HANDLE)
    {
     PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
     Print("SQL request: ", sql);
     DatabaseClose(db);
     return;
    }
//--- установим значение первого параметра запроса
  DatabaseBind(request, 0, _Symbol);
//--- запомним время старта перед добавлением баров в таблицу RATES
  start=GetMicrosecondCount();
  DatabaseTransactionBegin(db);
  int total=ArraySize(rates);
  bool request_error=false;
  for(int i=0; i<total; i++)
    {
     //--- устанавливаем значения остальных параметров перед добавлением записи
     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;
       }
     //--- если предыдущий вызов DatabaseBind() прошел успешно, то установим следующий
     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;
       }
```

© 2000-2025, MetaQuotes Ltd.

```
   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;
    }

   //--- выполним запрос на вставку записи и проверим на ошибку
   if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_MO
    {
     PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
     DatabaseFinalize(request);
     request_error=true;
```

© 2000-2025, MetaQuotes Ltd.

```
      break;
     }
    //--- сбросим запрос в начальное состояние перед следующим обновлением параметров
    if(!request_error && !DatabaseReset(request))
     {
      PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
      DatabaseFinalize(request);
      request_error=true;
      break;
     }
   } //--- закончили, прошли по всем барам

//--- как прошли транзакции?
  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);
   }
//--- сохраним таблицу RATES в CSV-файл
  string csv_filename=Symbol()+".csv";
  long saved=DatabaseExport(db, "SELECT * FROM RATES", csv_filename, DATABASE_EXPORT_H
  if(saved>0)
    Print("Table RATES saved in ", Symbol(), ".csv");
  else
    Print("DatabaseExport() failed. Error ", GetLastError());
//--- закроем файл с базой данных и сообщим об этом
  DatabaseClose(db);
  PrintFormat("Database: %s created and closed", filename);
 }
```

Смотри также

DatabasePrint, DatabaseImport

© 2000-2025, MetaQuotes Ltd.

DatabasePrint

Печатает таблицу или результат выполнения SQL-запроса в журнал экспертов.

```
long  DatabasePrint(
   int           database,          // хендл базы данных, полученный в DatabaseOpen
   const string  table_or_sql,      // таблица или SQL-запрос
   uint          flags              // комбинация флагов
   );
```

Параметры

database

[in] Хендл базы данных, который получен в DatabaseOpen().

table_or_sql

[in] Имя таблицы или текст SQL-запроса, результаты которого выведены в журнал экспертов.

flags

[in] Комбинация флагов, определяющая форматирование вывода. Флаги определены следующим образом:

DATABASE_PRINT_NO_HEADER - не выводить названия столбцов таблицы (имена полей)
DATABASE_PRINT_NO_INDEX - не выводить номера строк
DATABASE_PRINT_NO_FRAME - не выводить фрейм, разделяющий заголовок и данные
DATABASE_PRINT_STRINGS_RIGHT - выравнивать строки вправо.

Если flags=0, то выводятся столбцы и строки, заголовок и данные разделяются фреймом, строки выравниваются влево.

Возвращаемое значение

Возвращает количество экспортированных строк или -1 в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INTERNAL_ERROR (4001) - критическая ошибка исполняющей системы;
- ERR_NOT_ENOUGH_MEMORY (4004) - недостаточно памяти;
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;

Примечание

Если в журнал выводятся результаты запроса, то SQL-запрос должен начинаться с "SELECT" или "select". Другими словами, SQL-запрос не может изменять состояние базы данных, в противном случае DatabasePrint() завершится ошибкой.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string filename="departments.sqlite";
```

© 2000-2025, MetaQuotes Ltd.

```
//--- создадим или откроем базу данных в общей папке терминалов
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATAB
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }

//--- создадим таблицу COMPANY
   if(!CreateTableCompany(db))
     {
      DatabaseClose(db);
      return;
     }
//--- создадим таблицу DEPARTMENT
   if(!CreateTableDepartment(db))
     {
      DatabaseClose(db);
      return;
     }

//--- выведем список всех полей в таблицах COMPANY и 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;
     }
//--- выведем в лог таблицу 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;
     }
//--- текст запроса для объединения таблиц COMPANY и DEPARTMENT
   string request="SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT "
                  "ON COMPANY.ID = DEPARTMENT.EMP_ID";
//--- выведем результат объединения таблиц
   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;
     }
//--- закрываем базу данных
```

© 2000-2025, MetaQuotes Ltd.

```
   DatabaseClose(db);
  }
/*
Вывод:
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
*/
//+------------------------------------------------------------------+
//| Создает таблицу COMPANY                                          |
//+------------------------------------------------------------------+
bool CreateTableCompany(int database)
  {
//--- если таблица COMPANY существует, то удалим её
   if(DatabaseTableExists(database, "COMPANY"))
     {
      //--- удаляем таблицу
      if(!DatabaseExecute(database, "DROP TABLE COMPANY"))
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         Print("Failed to drop table COMPANY with code ", GetLastError());
         return(false);
        }
     }
//--- создаем таблицу 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);
     }

//--- вставляем данные в таблицу 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);
     }
//--- успех
   return(true);
  }
//+------------------------------------------------------------------+
//| Создает таблицу DEPARTMENT                                       |
//+------------------------------------------------------------------+
bool CreateTableDepartment(int database)
  {
//--- если таблица DEPARTMENT  существует, то удалим её
   if(DatabaseTableExists(database, "DEPARTMENT"))
     {
      //--- удаляем таблицу
      if(!DatabaseExecute(database, "DROP TABLE DEPARTMENT"))
        {
         Print("Failed to drop table DEPARTMENT  with code ", GetLastError());
         return(false);
        }
     }
//--- создаем таблицу DEPARTMENT
   if(!DatabaseExecute(database, "CREATE TABLE DEPARTMENT ("
                       "ID      INT PRIMARY KEY   NOT NULL,"
```

© 2000-2025, MetaQuotes Ltd.

```
                        "DEPT    CHAR(50)          NOT NULL,"
                        "EMP_ID  INT               NOT NULL);"))
      {
       Print("DB: create table DEPARTMENT failed with code ", GetLastError());
       return(false);
      }

//--- вставляем данные в таблицу 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);
      }
//--- успех
   return(true);
  }
//+------------------------------------------------------------------+
```

Смотри также

DatabaseExport, DatabaseImport

© 2000-2025, MetaQuotes Ltd.

DatabaseTableExists

Проверяет наличие таблицы в базе данных.

```
bool  DatabaseTableExists(
   int     database,      // хендл базы данных, полученный в DatabaseOpen
   string  table          // имя таблицы
   );
```

Параметры

database

[in] Хендл базы данных, который получен в DatabaseOpen().

table

[in] Имя таблицы.

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INVALID_PARAMETER (4003) - имя таблицы не задано (пустая строка или NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - ошибка конвертации запроса в UTF-8 строку;
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_EXECUTE (5124) - ошибка выполнения запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - таблицы не существует (не является ошибкой, нормальное завершение).

Смотри также

DatabasePrepare, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabaseExecute

Исполняет запрос к указанной базе данных.

```
bool  DatabaseExecute(
   int     database,      // хендл базы данных, полученный в DatabaseOpen
   string  sql            // SQL-запрос
   );
```

Параметры

database

[in] Хендл базы данных, который получен в DatabaseOpen().

sql

[in] SQL запрос.

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INTERNAL_ERROR (4001) - критическая ошибка исполняющей системы;
- ERR_INVALID_PARAMETER (4003) - параметр sql содержит пустую строку;
- ERR_NOT_ENOUGH_MEMORY (4004) - недостаточно памяти;
- ERR_WRONG_STRING_PARAMETER (5040) - ошибка конвертации запроса в UTF-8 строку;
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_EXECUTE (5124) - ошибка выполнения запроса.

Пример:

```
//--- статистика по символу
struct Symbol_Stats
  {
   string            name;             // имя символа
   int               trades;           // количество трейдов по символу
   double            gross_profit;     // общая прибыль по символу
   double            gross_loss;       // общий убыток по символу
   double            total_commission; // сумма комиссий по символу
   double            total_swap;       // сумма свопов по символу
   double            total_profit;     // общая прибыль без учета свопов и комиссий
   double            net_profit;       // чистая прибыль с учетом свопов и комиссий
   int               win_trades;       // количество прибыльных трейдов
   int               loss_trades;      // количество убыточных трейдов
   double            expected_payoff;  // матожидание трейда без учета свопов и комисс
   double            win_percent;      // процент выигрышных трейдов
   double            loss_percent;     // процент проигрышных трейдов
   double            average_profit;   // средняя прибыль
   double            average_loss;     // средний убыток
   double            profit_factor;    // профит-фактор
```

© 2000-2025, MetaQuotes Ltd.

```
  };

//--- статистика по Magic Number
struct Magic_Stats
  {
   long              magic;            // Magic Number советника
   int               trades;           // количество трейдов по символу
   double            gross_profit;     // общая прибыль по символу
   double            gross_loss;       // общий убыток по символу
   double            total_commission; // сумма комиссий по символу
   double            total_swap;       // сумма свопов по символу
   double            total_profit;     // общая прибыль без учета свопов и комиссий
   double            net_profit;       // чистая прибыль с учетом свопов и комиссий
   int               win_trades;       // количество прибыльных трейдов
   int               loss_trades;      // количество убыточных трейдов
   double            expected_payoff;  // матожидание трейда без учета свопов и комисс
   double            win_percent;      // процент выигрышных трейдов
   double            loss_percent;     // процент проигрышных трейдов
   double            average_profit;   // средняя прибыль
   double            average_loss;     // средний убыток
   double            profit_factor;    // профит-фактор
  };

//--- статистика по часу входа
struct Hour_Stats
  {
   char              hour_in;          // час входа в рынок
   int               trades;           // количество трейдов в этот час входа
   double            volume;           // объем трейдов в этот час входа
   double            gross_profit;     // общая прибыль в этот час входа
   double            gross_loss;       // общий убыток в этот час входа
   double            net_profit;       // чистая прибыль с учетом свопов и комиссий
   int               win_trades;       // количество прибыльных трейдов
   int               loss_trades;      // количество убыточных трейдов
   double            expected_payoff;  // матожидание трейда без учета свопов и комисс
   double            win_percent;      // процент выигрышных трейдов
   double            loss_percent;     // процент проигрышных трейдов
   double            average_profit;   // средняя прибыль
   double            average_loss;     // средний убыток
   double            profit_factor;    // профит-фактор
  };

int ExtDealsTotal=0;;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим имя файла
```

© 2000-2025, MetaQuotes Ltd.

```
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_stats.sqlite";
//--- открываем/создаем базу данных в общей папке терминалов
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- создадим таблицу DEALS
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
   PrintFormat("Торговая история насчитывает сделок: %d ", ExtDealsTotal);

//--- получим торговую статистику в разрезе символов
   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;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- получаем записи из результатов запроса
   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("");
//--- удалим запрос
   DatabaseFinalize(request);


//--- получим торговую статистику в разрезе советников по Magic Number
   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 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_trad
                           "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                           "   FROM DEALS "
```

© 2000-2025, MetaQuotes Ltd.

```
                              "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                              "   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;
//--- выводим записи
   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("");
//--- удалим запрос
   DatabaseFinalize(request);

//--- проверим что на счете используется хеджинг для учета открытых позиций
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
      //--- не можем преобразовать сделки в трейды простым способом через транзакции,
      DatabaseClose(db);
      return;
     }

//--- теперь создадим таблицу TRADES на основе таблицы DEALS
   if(!CreateTableTrades(db))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       DatabaseClose(db);
       return;
      }
//--- заполним через SQL-запрос таблицу TRADES на основе данных из DEALS
   if(DatabaseTableExists(db, "DEALS"))
      //--- заполним таблицу TRADES
      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;
        }

//--- получим торговую статистику в разрезе часа входа в рынок
   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 "
```

© 2000-2025, MetaQuotes Ltd.

```
                              "   GROUP BY HOUR_IN"
                              "   ) 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;
//--- выводим записи
   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("");
//--- удалим запрос
   DatabaseFinalize(request);

//--- close database
   DatabaseClose(db);
   return;
  }
/*
Торговая история насчитывает сделок: 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
```

© 2000-2025, MetaQuotes Ltd.

```
[5] "GBPJPY"      127     1943.43000  -1776.67000          -18.84000    -52.46000
[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
*/
//+------------------------------------------------------------------+
//| Создает таблицу DEALS                                            |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- если таблица DEALS уже есть, удалим её
   if(!DeleteTable(database, "DEALS"))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       return(false);
      }
//--- проверим наличие таблицы
   if(!DatabaseTableExists(database, "DEALS"))
      //--- создаем таблицу
      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);
        }
//---  запросим всю торговую историю
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- запросим историю сделок в указанном интервале
   HistorySelect(from_date, to_date);
   ExtDealsTotal=HistoryDealsTotal();
//--- внесем в таблицу сделки
   if(!InsertDeals(database))
      return(false);
//--- таблица успешно создана
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет из базы таблицу с указанным именем                       |
//+------------------------------------------------------------------+
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);
     }
//--- таблица успешно удалена
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Вносит сделки в таблицу базы данных                              |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- вспомогательные переменные
   ulong    deal_ticket;          // тикет сделки
   long     order_ticket;         // тикет ордера,по которому была совершена сделка
   long     position_ticket;      // ID позиции, к которой относится сделка
   datetime time;                 // время совершения сделки
   long     type ;                // тип сделки
   long     entry ;               // направление сделки
   string   symbol;               // по какому символу была сделка
   double   volume;               // объем операции
   double   price;                // цена
   double   profit;               // финансовый результат
   double   swap;                 // своп
   double   commission;           // комиссия
   long     magic;                // Magic number (ID советника)
   long     reason;               // причина или источник проведения сделки
   char     hour;                 // час совершения сделки
   MqlDateTime time_strusture;
//--- пройдем по всем сделкам и внесем их в базу данных
   bool failed=false;
   int deals=HistoryDealsTotal();
//--- заблокируем базу данных перед выполнением транзакций
   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;
      //--- внесем в таблицу каждую сделку через запрос
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIN
```

© 2000-2025, 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;
        }
     }
//--- проверим на наличие ошибок при выполнении транзакций
   if(failed)
     {
      //--- откатим все транзакции и разблокируем базу данных
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastErro
      return(false);
     }
//--- все транзакции прошли успешно - зафиксируем изменения и разблокируем базу данных
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Создает таблицу TRADES                                           |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- если таблица TRADES уже есть, удалим её
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- проверим наличие таблицы
   if(!DatabaseTableExists(database, "TRADES"))
      //--- создаем таблицу
      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);
```

© 2000-2025, MetaQuotes Ltd.

```
         }
//--- таблица успешно создана
   return(true);
  }
//+------------------------------------------------------------------+
```

Смотри также

DatabasePrepare, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabasePrepare

Создает хендл запроса, который затем может быть исполнен с помощью DatabaseRead().

```
int  DatabasePrepare(
   int     database,      // хендл базы данных, полученный в DatabaseOpen
   string  sql,           // SQL-запрос
           ...            // параметры запроса
   );
```

Параметры

database

[in] Хендл базы данных, который получен в DatabaseOpen().

sql

[in] SQL запрос, который может содержать автоподставляемые параметры с именами ?1,?2,...

...

[in] Автоматически подставляемые параметры запроса.

Возвращаемое значение

При успешной выполнении возвращает хэндл на SQL-запрос, в противном случае возвращает INVALID_HANDLE. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INVALID_PARAMETER (4003) - путь к файлу базы содержит пустую строку или выставлена несовместимая комбинация флагов;
- ERR_NOT_ENOUGH_MEMORY (4004) - недостаточно памяти;
- ERR_WRONG_STRING_PARAMETER (5040) - ошибка конвертации запроса в UTF-8 строку;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - превышено максимально допустимое количество объектов Database;
- ERR_DATABASE_PREPARE (5125) - ошибка создания запроса.

Примечание

Функция DatabasePrepare() не выполняет запрос к базе данных. Её назначение - проверить параметры запроса на корректность и вернуть по результатам данной проверки хендл для выполнения SQL-запроса. Сам запрос будет выполнен при первом вызове DatabaseRead().

Пример:

```
//--- структура для хранения сделки
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
```

© 2000-2025, 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
  };
//--- структура для хранения трейда - порядок членов соответствует позиции в терминале
struct Trade
  {
   datetime          time_in;          // время входа
   ulong             ticket;           // ID позиции
   char              type;             // покупка или продажа
   double            volume;           // объем
   string            symbol;           // символ
   double            price_in;         // цена входа
   datetime          time_out;         // время выхода
   double            price_out;        // цена выхода
   double            commission;       // комиссия за вход и выход
   double            swap;             // своп
   double            profit;           // прибыль или убыток
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим имя файла
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_trades.sqlite"
//--- открываем/создаем базу данных в общей папке терминалов
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- создадим таблицу DEALS
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  запросим всю торговую историю
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- запросим историю сделок в указанном интервале
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("Торговая история насчитывает сделок: %d ", deals_total);
//--- внесем в таблицу сделки
   if(!InsertDeals(db))
      return;
//--- покажем первые 10 сделок
   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;
     }
//--- выведем сделки на печать
   if(i>0)
     {
      ArrayResize(deals, i);
      PrintFormat("Первые %d сделок:", i);
      ArrayPrint(deals);
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- проверим что на счете используется хеджинг для учета открытых позиций
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
      //--- не можем преобразовать сделки в трейды простым способом через транзакции,
      DatabaseClose(db);
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- теперь создадим таблицу TRADES на основе таблицы DEALS
   if(!CreateTableTrades(db))
     {
      DatabaseClose(db);
      return;
     }
//--- заполним через SQL-запрос таблицу TRADES на основе данных из DEALS
   ulong start=GetMicrosecondCount();
   if(DatabaseTableExists(db, "DEALS"))
      //--- заполним таблицу TRADES
      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;

//--- покажем первые 10 сделок
   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)
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
      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;
     }
//--- выведем трейды на печать
   if(i>0)
     {
      ArrayResize(trades, i);
      PrintFormat("\r\nПервые %d трейдов:", i);
      ArrayPrint(trades);
      PrintFormat("Заполнение таблицы TRADES заняло %.2f миллисекунд",double(transacti
     }
//--- удалим запрос после использования
   DatabaseFinalize(request);

//--- закрываем базу данных
   DatabaseClose(db);
  }
/*
Результат:
   Торговая история насчитывает сделок: 2741
   Первые 10 сделок:
       [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 "G
   [9] 34435154       51450479          51450479 2019.09.06 10:30:00      1        0 "E

   Первые 10 трейдов:
                [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:
   [4] 2019.09.06 08:00:00 51448053      1  0.10000 "USDCHF"    0.98701 2019.09.06 18:
```

© 2000-2025, MetaQuotes Ltd.

```
   [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:
   Заполнение таблицы TRADES заняло 12.51 milliseconds
*/
//+------------------------------------------------------------------+
//| Создает таблицу DEALS                                            |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- если таблица DEALS уже есть, удалим её
   if(!DeleteTable(database, "DEALS"))
     {
      return(false);
     }
//--- проверим наличие таблицы
   if(!DatabaseTableExists(database, "DEALS"))
      //--- создаем таблицу
      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);
        }
//--- таблица успешно создана
   return(true);
  }
//+------------------------------------------------------------------+
//| Удаляет из базы таблицу с указанным именем                       |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("Failed to drop the DEALS table with code ", GetLastError());
      return(false);
     }
//--- таблица успешно удалена
   return(true);
  }
//+------------------------------------------------------------------+
//| Вносит сделки в таблицу базы данных                              |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- вспомогательные переменные
   ulong    deal_ticket;          // тикет сделки
   long     order_ticket;         // тикет ордера,по которому была совершена сделка
   long     position_ticket;      // ID позиции, к которой относится сделка
   datetime time;                 // время совершения сделки
   long     type ;                // тип сделки
   long     entry ;               // направление сделки
   string   symbol;               // по какому символу была сделка
   double   volume;               // объем операции
   double   price;                // цена
   double   profit;               // финансовый результат
   double   swap;                 // своп
   double   commission;           // комиссия
   long     magic;                // Magic number (ID советника)
   long     reason;               // причина или источник проведения сделки
//--- пройдем по всем сделкам и внесем их в базу данных
   bool failed=false;
   int deals=HistoryDealsTotal();
//--- заблокируем базу данных перед выполнением транзакций
   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);
      //--- внесем в таблицу каждую сделку через запрос
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
```

© 2000-2025, 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;
        }
     }
//--- проверим на наличие ошибок при выполнении транзакций
   if(failed)
     {
      //--- откатим все транзакции и разблокируем базу данных
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code %d", __FUNCTION__, GetLastEr
      return(false);
     }
//--- все транзакции прошли успешно - зафиксируем изменения и разблокируем базу данных
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Создает таблицу TRADES                                           |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- если таблица TRADES уже есть, удалим её
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- проверим наличие таблицы
   if(!DatabaseTableExists(database, "TRADES"))
      //--- создаем таблицу
      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);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- таблица успешно создана
   return(true);
  }
//+------------------------------------------------------------------+
```

Смотри также

DatabaseExecute, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabaseReset

Сбрасывает запрос в начальное состояние, как после вызова DatabasePrepare().

```
int  DatabaseReset(
   int  request      // хендл запроса, полученный в DatabasePrepare
   );
```

Параметры

request

[in] Хендл запроса, который получен в DatabasePrepare().

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- коды ошибок SQLite, начиная с ERR_DATABASE_ERROR(5601).

Примечание

Функция DatabaseReset() предназначена для многократного исполнения запроса с разными значениям параметров. Например, при массовом добавлении данных в таблицу с помощью команды INSERT для каждой записи необходимо сформировать собственный набор значений каждого поля.

В отличие от DatabasePrepare(), вызов DatabaseReset() не производит компиляцию строки с SQL-командами в новый запрос, поэтому DatabaseReset() выполняется значительно быстрее, чем DatabasePrepare().

DatabaseReset() используется совместно с функциями DatabaseBind() и/или DatabaseBindArray(), если необходимо изменить значения параметров запроса после выполнения DatabaseRead(). Это означает, что перед установкой новых значений параметров запроса (перед блоком вызовов DatabaseBind/DatabaseBindArray) необходимо вызвать DatabaseReset() для сброса его в начальное состояние. Сам параметризованный запрос должен быть создан с помощью DatabasePrepare().

DatabaseReset() как и DatabasePrepare() не выполняет запроса к базе данных. Непосредственное выполнение запроса происходит при вызове DatabaseRead() или DatabaseReadBind().

Вызов DatabaseReset() не приводит к сбросу значений параметров в запросе, если они были выставлены вызовом DatabaseBind()/DatabaseBindArray(), то есть параметры сохраняют свои значения. Таким образом, можно менять значение только одного параметра - нет необходимости после вызова DatabaseReset() прописывать заново все параметры запроса.

В DatabaseReset() нельзя передавать хендл запроса, который был перед этим удален с помощью DatabaseFinalize(). Это приведёт к ошибке.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- создадим или откроем базу данных
   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");
//--- если таблица SYMBOLS существует, то удалим её
   if(DatabaseTableExists(db, "SYMBOLS"))
     {
      //--- удаляем таблицу
      if(!DatabaseExecute(db, "DROP TABLE SYMBOLS"))
        {
         Print("Failed to drop table SYMBOLS with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- создаем таблицу 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;
     }
//--- покажем список всех полей в таблице SYMBOLS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(SYMBOLS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(SYMBOLS)\") failed, error code=%d
      DatabaseClose(db);
      return;
     }

//--- создадим параметризованный запрос добавления символов в таблицу SYMBOLS
   string sql="INSERT INTO SYMBOLS (NAME,DESCRIPTION,PATH,SPREAD,POINT,DIGITS,JSON)"
              " VALUES (?1,?2,?3,?4,?5,?6,?7);"; // параметры запроса
   int request=DatabasePrepare(db, sql);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- пройдем по всем символам и добавим их в таблицу SYMBOLS
   int symbols=SymbolsTotal(false);
   bool request_error=false;
   DatabaseTransactionBegin(db);
   for(int i=0; i<symbols; i++)
     {
      //--- устанавливаем значения параметров перед добавлением символа
      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;
        }
      //--- если предыдущий вызов DatabaseBind() прошел успешно, то установим следующи
      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)))
```

© 2000-2025, MetaQuotes Ltd.

```
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         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;
        }

      //--- выполним запрос на вставку записи и проверим на ошибку
      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);
      //--- сбросим запрос в начальное состояние перед следующим обновлением параметро
      if(!DatabaseReset(request))
        {
         PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- закончили, прошли по всем символам

//--- как прошли транзакции?
   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);
     }

//--- сохраним таблицу SYMBOLS в CSV-файл
   string csv_filename="symbols.csv";
   if(DatabaseExport(db, "SELECT * FROM SYMBOLS", csv_filename,
```

© 2000-2025, MetaQuotes Ltd.

```
                     DATABASE_EXPORT_HEADER|DATABASE_EXPORT_INDEX|DATABASE_EXPORT_QUOT
      Print("Database: table SYMBOLS saved in ", csv_filename);
   else
      Print("Database: DatabaseExport(\"SELECT * FROM SYMBOLS\") failed with code", Ge

//--- закроем файл с базой данных и сообщим об этом
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
//+------------------------------------------------------------------+
//| Возвращает спецификацию символа в виде JSON                      |
//+------------------------------------------------------------------+
string GetSymBolAsJson(string symbol)
  {
//--- отступы
   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(SymbolInfoIntege
               "\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(syr
               "\n"+indent3+"\"ContractSize\":\""+StringFormat("%G",(SymbolInfoDouble
               "\n"+indent3+"\"StopsLevel\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"VolumeMin\":\""+StringFormat("%G",(SymbolInfoDouble(syr
               "\n"+indent3+"\"VolumeMax\":\""+StringFormat("%G",(SymbolInfoDouble(syr
               "\n"+indent3+"\"VolumeStep\":\""+StringFormat("%G",(SymbolInfoDouble(sy
```

© 2000-2025, MetaQuotes Ltd.

```
                "\n"+indent3+"\"VolumeLimit\":\""+StringFormat("%G",(SymbolInfoDouble(s
                "\n"+indent3+"\"SwapMode\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent3+"\"SwapLong\":\""+StringFormat("%G",(SymbolInfoDouble(symb
                "\n"+indent3+"\"SwapShort\":\""+StringFormat("%G",(SymbolInfoDouble(syr
                "\n"+indent3+"\"Swap3Day\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent2+"}"+
                "\n"+indent1+"]"+
                "\n}";
   return(json);
  }
//+------------------------------------------------------------------+
//| Формирует отступ из пробелов                                     |
//+------------------------------------------------------------------+
string Indent(const int number, const int characters=3)
  {
   int length=number*characters;
   string indent=NULL;
   StringInit(indent, length, ' ');
   return indent;
  }
/*
  Результат:
  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
*/
```

Смотри также

DatabasePrepare, DatabaseBind, DatabaseBindArray, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabaseBind

Устанавливает значение параметра в запросе.

```
bool  DatabaseBind(
   int   request,       // хендл запроса, созданного в DatabasePrepare
   int   index,         // индекс параметра в запросе
   T     value          // значение параметра простого типа
   );
```

Параметры

request

[in] Хендл запроса, созданного в DatabasePrepare().

index

[in] Индекс параметра в запросе, для которого нужно установить значение. Нумерация начинается с нуля.

value

[in] Значение параметра, которое необходимо установить. Разрешенные типы: bool, char, uchar, short, ushart, int, uint, color, datetime, long, ulong, float, double, string.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INVALID_PARAMETER (4003) - неподдерживаемый тип;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_NOT_READY (5128) - нельзя использовать функцию для запроса в данный момент. Запрос выполняется или уже завершён, требуется вызвать DatabaseReset().

Примечание

Функцию следует использовать в случае, когда SQL запрос содержит параметризируемые значения "?" или "?N", где N означает номер параметра (начиная с единицы). При этом индексация параметров в DatabaseBind() начинается с нуля.

Например:

```
INSERT INTO table VALUES (?,?,?)
SELECT * FROM table WHERE id=?
```

Функцию можно вызывать сразу после того, как параметризованный запрос был создан в DatabasePrepare(), либо после сброса запроса в начальное состояние с помощью DatabaseReset().

Используйте эту функцию совместно с DatabaseReset(), чтобы выполнить запрос нужное количество раз с разными значениями параметров.

Функция предназначена для работы с параметрами простых типов. Если параметру требуется сопоставить массив, используйте функцию DatabaseBindArray().

© 2000-2025, MetaQuotes Ltd.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlTick ticks[];
//--- запомним время старта перед получением тиков
   uint start=GetTickCount();
//--- запросим тиковую историю за сутки
   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
     {
      //--- сколько тиков и за какое время получили
      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)));
     }

//--- составим имя файла для хранения базы данных
   string filename=_Symbol+" "+TimeToString(datetime(from/1000))+" - "+TimeToString(da
   StringReplace(filename, ":", "."); // символ ":" запрещен в названиях файлов
//--- открываем/создаем базу данных в общей папке терминалов
   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");

//--- создаем таблицу 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,"
```

© 2000-2025, MetaQuotes Ltd.

```
                        "VOLUME_REAL        REAL);"))
     {
      Print("DB: ", filename, " create table TICKS failed with code ", GetLastError())
      DatabaseClose(db);
      return;
     }
//--- покажем список всех полей в таблице TICKS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(TICKS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(TICKS)\") failed, error code=%d a
      DatabaseClose(db);
      return;
     }
//--- создадим параметризованный запрос добавления тиков в таблицу TICKS
   string sql="INSERT INTO TICKS (SYMBOL,TIME,BID,ASK,LAST,VOLUME,TIME_MSC,VOLUME_REAI
              " VALUES (?1,?2,?3,?4,?5,?6,?7,?8)"; // параметры запроса
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }
//--- установим значение первого параметра запроса
   DatabaseBind(request, 0, _Symbol);
//--- запомним время старта перед добавлением тиков в таблицу TICKS
   start=GetTickCount();
   DatabaseTransactionBegin(db);
   int total=ArraySize(ticks);
   bool request_error=false;
   for(int i=0; i<total; i++)
     {
      //--- устанавливаем значения остальных параметров перед добавлением записи
      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;
        }
      //--- если предыдущий вызов DatabaseBind() прошел успешно, то установим следующ
      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;
```

© 2000-2025, 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;
      }

   //--- выполним запрос на вставку записи и проверим на ошибку
   if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_
      {
       PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
   //--- сбросим запрос в начальное состояние перед следующим обновлением параметро
   if(!request_error && !DatabaseReset(request))
      {
       PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- закончили, прошли по всем тикам

//--- как прошли транзакции?
   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);
     }

//--- закроем файл с базой данных и сообщим об этом
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
/*
  Результат:
  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
*/
```

Смотри также

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBindArray

© 2000-2025, MetaQuotes Ltd.

DatabaseBindArray

Устанавливает массив в качестве значения параметра.

```
bool  DatabaseBind(
   int   request,       // хендл запроса, созданного в DatabasePrepare
   int   index,         // индекс параметра в запросе
   T&    array[]        // значение параметра в виде массива
   );
```

Параметры

request

[in] Хендл запроса, созданного в DatabasePrepare().

index

[in] Индекс параметра в запросе, для которого нужно установить значение. Нумерация начинается с нуля.

array[]

[in] Массив, который необходимо установить в качестве значения параметра запроса.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INVALID_PARAMETER (4003) - неподдерживаемый тип;
- ERR_ARRAY_BAD_SIZE (4011) - размер массива в байтах превышает значение INT_MAX;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_NOT_READY (5128) - нельзя использовать функцию для запроса в данный момент (Запрос уже выполняется или завершён, требуется вызвать DatabaseReset).

Примечание

Функцию следует использовать в случае, когда SQL запрос содержит параметризируемые значения "?" или "?N", где N означает номер параметра (начиная с единицы). При этом индексация параметров в DatabaseBindArray() начинается с нуля.

Например:

```
INSERT INTO table VALUES (?,?,?)
```

Функцию можно вызывать сразу после того, как параметризованный запрос был создан в DatabasePrepare(), либо после сброса запроса в начальное состояние с помощью DatabaseReset().

Используйте эту функцию совместно с DatabaseReset(), чтобы выполнить запрос нужное количество раз с разными значениями параметров.

Пример:

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

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- откроем диалог для выбора файлов с расширением DAT
   string selected_files[];
   if(!FileSelectDialog("Выберите файлы для загрузки", NULL,
                        "Data files (*.dat)|*.dat|All files (*.*)|*.*",
                        FSD_ALLOW_MULTISELECT, selected_files, "tester.dat")>0)
     {
      Print("Files not selected. Exit");
      return;
     }
//--- получим размеры файлов
   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];
        }
     }
//--- проверим общий размер файлов
   if(total_size==0)
     {
      PrintFormat("Total files size is 0. Exit");
      return;
     }

//--- создадим или откроем базу данных в общей папке терминалов
   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");
//--- если таблица FILES существует, то удалим её
   if(DatabaseTableExists(db, "FILES"))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- удаляем таблицу
       if(!DatabaseExecute(db, "DROP TABLE FILES"))
         {
          Print("Failed to drop table FILES with code ", GetLastError());
          DatabaseClose(db);
          return;
         }
      }
//--- создадим таблицу 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;
      }
//--- покажем список всех полей в таблице FILES
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(FILES)", 0)<0)
      {
       PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(FILES)\") failed, error code=%d a
       DatabaseClose(db);
       return;
      }

//--- создадим параметризованный запрос добавления файлов в таблицу FILES
   string sql="INSERT INTO FILES (NAME,SIZE,PERCENT_SIZE,DATA)"
              " VALUES (?1,?2,?3,?4);"; // параметры запроса
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
      {
       PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
       Print("SQL request: ", sql);
       DatabaseClose(db);
       return;
      }

//--- пройдем по всем файлам и добавим их в таблицу FILES
   bool request_error=false;
   DatabaseTransactionBegin(db);
   int count=0;
   uint size;
   for(int i=0; i<files; i++)
      {
       if(filehandle[i]!=INVALID_HANDLE)
         {
```

© 2000-2025, MetaQuotes Ltd.

```
      char data[];
      size=FileReadArray(filehandle[i], data);
      if(size==0)
        {
         PrintFormat("FileReadArray(%s) failed with code %d", selected_files[i], Ge
         continue;
        }

      count++;
      //--- устанавливаем значения параметров перед добавлением файла в таблицу
      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;
        }
      //--- выполним запрос на вставку записи и проверим на ошибку
      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);
      //--- сбросим запрос в начальное состояние перед следующим обновлением параме
      if(!DatabaseReset(request))
        {
         PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
         DatabaseFinalize(request);
```

© 2000-2025, MetaQuotes Ltd.

```
            request_error=true;
            break;
           }
        }
     }
//--- как прошли транзакции?
   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);
     }

//--- закроем файл с базой данных и сообщим об этом
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
```

Смотри также

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBind

© 2000-2025, MetaQuotes Ltd.

DatabaseRead

Выполняет переход к следующей записи в результате запроса.

```
bool  DatabaseRead(
   int  request      // хендл запроса, полученный в DatabasePrepare
   );
```

Параметры

request

[in] Хендл запроса, который получен в DatabasePrepare().

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INVALID_PARAMETER (4003) - имя таблицы не задано (пустая строка или NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - ошибка конвертации запроса в UTF-8 строку;
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_EXECUTE (5124) - ошибка выполнения запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - таблицы не существует (не является ошибкой, нормальное завершение).

Смотри также

DatabasePrepare, DatabaseReadBind

© 2000-2025, MetaQuotes Ltd.

DatabaseReadBind

Выполняет переход к следующей записи и читает из неё данные в структуру.

```
bool  DatabaseReadBind(
   int    request,           // хендл запроса, созданного в DatabasePrepare
   void&  struct_object      // ссылка на структуру для чтения записи
   );
```

Параметры

request

[in] Хендл запроса, созданного в DatabasePrepare().

struct_object

[out] Ссылка на структуру, в которую будут прочитаны данные из текущей записи. Структура должна иметь в качестве членов только числовые типы и/или строки (массивы не разрешены) и не может быть наследником.

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INVALID_PARAMETER (4003) - имя таблицы не задано (пустая строка или NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - ошибка конвертации запроса в UTF-8 строку;
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_EXECUTE (5124) - ошибка выполнения запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - таблицы не существует (не является ошибкой, нормальное завершение).

Примечание

Количество полей в структуре *struct_object* не может превышать DatabaseColumnsCount(). Если количество полей в структуре *struct_object* меньше количества полей в записи, то будет произведено частичное чтение. Оставшиеся данные можно будет получить явным образом с помощью соответствующих функций DatabaseColumnText(), DatabaseColumnInteger() и так далее.

Пример:

```
struct Person
  {
   int               id;
   string            name;
   int               age;
   string            address;
   double            salary;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   string filename="company.sqlite";
//--- создадим или откроем базу данных в общей папке терминалов
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATAB
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- если таблицыа COMPANY существует, то удалим её
   if(DatabaseTableExists(db, "COMPANY"))
     {
      //--- удаляем таблицу
      if(!DatabaseExecute(db, "DROP TABLE COMPANY"))
        {
         Print("Failed to drop table COMPANY with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- создаем таблицу
   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;
     }

//--- вставляем данные в таблицу
   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;
     }

//--- создадим запрос и получим хендл на него
   int request=DatabasePrepare(db, "SELECT * FROM COMPANY WHERE SALARY>15000");
   if(request==INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("DB: ", filename, " request failed with code ", GetLastError());
       DatabaseClose(db);
       return;
      }
//--- распечатаем все записи с зарплатой больше 15000
   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
//--- удалим запрос после использования
   DatabaseFinalize(request);

   Print("Some statistics:");
//--- подготовим новый запрос о сумме зарплат
   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);
     }
//--- удалим запрос после использования
   DatabaseFinalize(request);

//--- подготовим новый запрос о средней зарплате
   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);
     }
//--- удалим запрос после использования
   DatabaseFinalize(request);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- закроем базу
   DatabaseClose(db);
  }
//+------------------------------------------------------------------
/*
Результат выполнения:
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
*/
```

Смотри также

DatabasePrepare, DatabaseRead

© 2000-2025, MetaQuotes Ltd.

DatabaseFinalize

Удаляет запрос, созданный в DatabasePrepare().

```
void  DatabaseFinalize(
   int  request      // хендл запроса, полученный в DatabasePrepare
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

Возвращаемое значение

Нет.

Примечание

В случае навалидного хендла функция выставит ошибку ERR_DATABASE_INVALID_HANDLE. Проверить ошибку можно с помощью GetLastError().

Смотри также

DatabasePrepare, DatabaseExecute

© 2000-2025, MetaQuotes Ltd.

DatabaseTransactionBegin

Начинает выполнение транзакции.

```
bool  DatabaseTransactionBegin(
   int  database      // хендл базы данных, полученный в DatabaseOpen
   );
```

Параметры

database

[in] Хендл базы данных, полученный в DatabaseOpen().

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INTERNAL_ERROR (4001) - критическая ошибка исполняющей системы;
- ERR_INVALID_PARAMETER (4003) - параметр sql содержит пустую строку;
- ERR_NOT_ENOUGH_MEMORY (4004) - недостаточно памяти;
- ERR_WRONG_STRING_PARAMETER (5040) - ошибка конвертации запроса в UTF-8 строку;
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_EXECUTE (5124) - ошибка выполнения запроса.

Примечание

Функцию DatabaseTransactionBegin() необходимо вызывать перед выполнением транзакции. Любая транзакция должна начинаться вызовом DatabaseTransactionBegin() и заканчиваться вызовом DatabaseTransactionCommit().

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- создадим имя файла
   string filename=AccountInfoString(ACCOUNT_SERVER) +"_"+IntegerToString(AccountInfo
//--- открываем/создаем базу данных в общей папке терминалов
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- если таблица DEALS уже есть, удалим её
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- создадим таблицу DEALS
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  запросим всю торговую историю
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- запросим историю сделок в указанном интервале
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("Торговая история насчитывает сделок: %d ", deals_total);

//--- замерим скорость выполнения транзакций с использованием DatabaseTransactionBegin
   ulong start=GetMicrosecondCount();
   bool fast_transactions=true;
   InsertDeals(db, fast_transactions);
   double fast_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit

//--- удалим таблицу DEALS - будем создавать её заново
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
//--- создадим заново таблицу DEALS
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }

//--- сделаем тест заново без использования DatabaseTransactionBegin/DatabaseTransacti
   fast_transactions=false;
   start=GetMicrosecondCount();
   InsertDeals(db, fast_transactions);
   double slow_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit
//--- сообщим выигрыш по времени
   PrintFormat("Использование DatabaseTransactionBegin/DatabaseTransactionCommit дало
//--- закрываем базу данных
   DatabaseClose(db);
  }
/*
Результат:
   Торговая история насчитывает сделок: 2737
   Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit: time=48.5 r
```

© 2000-2025, MetaQuotes Ltd.

```
   Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit: time=25818.
   Использование DatabaseTransactionBegin/DatabaseTransactionCommit дало ускорение в 5
*/
//+------------------------------------------------------------------+
//| Удаляет из базы таблицу с указанным именем                       |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop table with code ", GetLastError());
      return(false);
     }
//--- таблица успешно удалена
   return(true);
  }
//+------------------------------------------------------------------+
//| Создает таблицу DEALS                                            |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- проверим наличие таблицы
   if(!DatabaseTableExists(database, "DEALS"))
      //--- создаем таблицу
      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 table DEALS failed with code ", GetLastError());
         return(false);
        }
//--- таблица успешно создана
   return(true);
  }
//+------------------------------------------------------------------+
//| Вносит сделки в таблицу базы данных                              |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool InsertDeals(int database, bool begintransaction=true)
  {
//--- вспомогательные переменные
   ulong    deal_ticket;          // тикет сделки
   long     order_ticket;         // тикет ордера,по которому была совершена сделка
   long     position_ticket;      // ID позиция, к которой относится сделка
   datetime time;                 // время совершения сделки
   long     type ;                // тип сделки
   long     entry ;               // направление сделки
   string   symbol;               // по какому символу была сделка
   double   volume;               // объем операции
   double   price;                // цена
   double   profit;               // финансовый результат
   double   swap;                 // своп
   double   commission;           // комиссия
   long     magic;                // Magic number
   long     reason;               // причина или источник проведения сделки
//--- пройдем по всем сделкам и внесем в базу данных
   bool failed=false;
   int deals=HistoryDealsTotal();
//--- если используется быстрый способ проведения транзакций
   if(begintransaction)
     {
      //--- заблокируем базу данных перед выполнением транзакций
      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);
      //--- внесем каждую сделку через запрос
      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
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- проверим на наличие ошибок при выполнении транзакций
   if(failed)
     {
      //--- если используется быстрый способ проведения транзакций
      if(begintransaction)
        {
         //--- откатим все транзакции и разблокируем базу данных
         DatabaseTransactionRollback(database);
        }
      Print("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastError());
      return(false);
     }
//--- если используется быстрый способ проведения транзакций
   if(begintransaction)
     {
      //--- все транзакции прошли успешно - зафиксируем изменения и разблокируем базу
      DatabaseTransactionCommit(database);
     }
//--- успешное завершение
   return(true);
  }
//+------------------------------------------------------------------+
```

Смотри также

DatabaseExecute, DatabasePrepare, DatabaseTransactionCommit, DatabaseTransactionRollback

© 2000-2025, MetaQuotes Ltd.

DatabaseTransactionCommit

Завершает выполнение транзакции.

```
bool  DatabaseTransactionCommit(
   int  database      // хендл базы данных, полученный в DatabaseOpen
   );
```

Параметры

database

[in] Хендл базы данных, полученный в DatabaseOpen().

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INTERNAL_ERROR (4001) - критическая ошибка исполняющей системы;
- ERR_INVALID_PARAMETER (4003) - параметр sql содержит пустую строку;
- ERR_NOT_ENOUGH_MEMORY (4004) - недостаточно памяти;
- ERR_WRONG_STRING_PARAMETER (5040) - ошибка конвертации запроса в UTF-8 строку;
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_EXECUTE (5124) - ошибка выполнения запроса.

Примечание

Функция DatabaseTransactionCommit() завершает все транзакции, которые были выполнены после вызова функции DatabaseBeginTransaction(). Любая транзакция для успешного завершения должна начинаться вызовом DatabaseTransactionBegin() и заканчиваться вызовом DatabaseTransactionCommit().

Смотри также

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionRollback

© 2000-2025, MetaQuotes Ltd.

DatabaseTransactionRollback

Выполняет откат транзакций.

```
bool  DatabaseTransactionRollback(
   int  database      // хендл базы данных, полученный в DatabaseOpen
   );
```

Параметры

database

[in] Хендл базы данных, полученный в DatabaseOpen().

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_INTERNAL_ERROR (4001) - критическая ошибка исполняющей системы;
- ERR_INVALID_PARAMETER (4003) - параметр sql содержит пустую строку;
- ERR_NOT_ENOUGH_MEMORY (4004) - недостаточно памяти;
- ERR_WRONG_STRING_PARAMETER (5040) - ошибка конвертации запроса в UTF-8 строку;
- ERR_DATABASE_INTERNAL (5120) - внутренняя ошибка базы данных;
- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл базы данных;
- ERR_DATABASE_EXECUTE (5124) - ошибка выполнения запроса.

Примечание

Вызов DatabaseTransactionRollback() отменяет все транзакции, которые были выполнены после вызова функции DatabaseTransactionBegin(). Функция DatabaseTransactionRollback() необходима для отката изменений в базе данных, если в процессе выполнения транзакции возникли ошибки.

Смотри также

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionCommit

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnsCount

Получает количество полей в запросе.

```
int  DatabaseColumnsCount(
   int  request      // хендл запроса, полученный в DatabasePrepare
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

Возвращаемое значение

Количество полей или -1 в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл запроса.

Примечание

Для получения количества полей запроса, созданного в DatabasePrepare(), нет необходимости вызывать функцию DatabaseRead(). Для остальных функций вида DatabaseColumnXXX() предварительно требуется вызвать DatabaseRead().

Смотри также

DatabasePrepare, DatabaseFinalize, DatabaseClose

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnName

Получает имя поля по номеру.

```
bool  DatabaseColumnName(
   int       request,      // хендл запроса, полученный в DatabasePrepare
   int       column,       // номер поля в запросе
   string&   name          // ссылка на переменную для получения имени поля
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

column

[in] Номер поля в запросе. Нумерация полей начинается с нуля и не может превышать значение DatabaseColumnsCount() - 1.

name

[out] Переменная для записи имени поля.

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - индекс column больше значения DatabaseColumnsCount() -1.

Примечание

Значение можно получить только в том случае, если предварительно для запроса *request* был сделан хотя бы один вызов DatabaseRead().

Смотри также

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnType

Получает тип поля по номеру.

```
ENUM_DATABASE_FIELD_TYPE  DatabaseColumnType(
   int  request,     // хендл запроса, полученный в DatabasePrepare
   int  column       // номер поля в запросе
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

column

[in] Номер поля в запросе. Нумерация полей начинается с нуля и не может превышать значение DatabaseColumnsCount() - 1.

Возвращаемое значение

Возвращает тип поля из перечисления ENUM_DATABASE_FIELD_TYPE. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - индекс column больше значения DatabaseColumnsCount() -1.

Примечание

Значение можно получить только в том случае, если предварительно для запроса *request* был сделан хотя бы один вызов DatabaseRead().

ENUM_DATABASE_FIELD_TYPE

Идентификатор	Описание
DATABASE_FIELD_TYPE_INVALID	Ошибка получения типа, код ошибки можно получить при помощи int GetLastError()
DATABASE_FIELD_TYPE_INTEGER	Целочисленный тип
DATABASE_FIELD_TYPE_FLOAT	Вещественный тип
DATABASE_FIELD_TYPE_TEXT	Строковой тип
DATABASE_FIELD_TYPE_BLOB	Бинарный тип
DATABASE_FIELD_TYPE_NULL	Специальный тип NULL

Смотри также

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnSize

Получает размер поля в байтах.

```
int  DatabaseColumnSize(
   int  request,     // хендл запроса, полученный в DatabasePrepare
   int  column       // номер поля в запросе
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

column

[in] Номер поля в запросе. Нумерация полей начинается с нуля и не может превышать значение DatabaseColumnsCount() - 1.

Возвращаемое значение

В случае успеха возвращает размер поля в байтах, иначе -1. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - индекс column больше значения DatabaseColumnsCount() -1.

Примечание

Значение можно получить только в том случае, если предварительно для запроса *request* был сделан хотя бы один вызов DatabaseRead().

Смотри также

DatabasePrepare, DatabaseColumnBlob, DatabaseColumnsCount, DatabaseColumnName, DatabaseColumnType

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnText

Получает из текущей записи значение поля в виде строки.

```
bool  DatabaseColumnText(
   int      request,     // хендл запроса, полученный в DatabasePrepare
   int      column,      // номер поля в запросе
   string&  value        // ссылка на переменную для получения значения
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

column

[in] Номер поля в запросе. Нумерация полей начинается с нуля и не может превышать значение DatabaseColumnsCount() - 1.

value

[out] Ссылка на переменную для записи значения поля.

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - индекс column больше значения DatabaseColumnsCount() -1.

Примечание

Значение можно получить только в том случае, если предварительно для запроса request был сделан хотя бы один вызов DatabaseRead().

Для чтения значения из следующей записи нужно предварительно вызвать DatabaseRead().

Смотри также

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnInteger

Получает из текущей записи значение типа int.

```
bool  DatabaseColumnInteger(
   int   request,     // хендл запроса, полученный в DatabasePrepare
   int   column,      // номер поля в запросе
   int&  value        // ссылка на переменную для получения значения
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

column

[in] Номер поля в запросе. Нумерация полей начинается с нуля и не может превышать значение DatabaseColumnsCount() - 1.

value

[out] Ссылка на переменную для записи значения поля.

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - индекс column больше значения DatabaseColumnsCount() -1.

Примечание

Значение можно получить только в том случае, если предварительно для запроса request был сделан хотя бы один вызов DatabaseRead().

Для чтения значения из следующей записи нужно предварительно вызвать DatabaseRead().

Смотри также

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnLong

Получает из текущей записи значение типа long.

```
bool  DatabaseColumnLong(
   int     request,      // хендл запроса, полученный в DatabasePrepare
   int     column,       // номер поля в запросе
   long&   value         // ссылка на переменную для получения значения
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

column

[in] Номер поля в запросе. Нумерация полей начинается с нуля и не может превышать значение DatabaseColumnsCount() - 1.

value

[out] Ссылка на переменную для записи значения поля.

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - индекс column больше значения DatabaseColumnsCount() -1.

Примечание

Значение можно получить только в том случае, если предварительно для запроса request был сделан хотя бы один вызов DatabaseRead().

Для чтения значения из следующей записи нужно предварительно вызвать DatabaseRead().

Смотри также

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnDouble

Получает из текущей записи значение типа double.

```
bool  DatabaseColumnDouble(
   int      request,     // хендл запроса, полученный в DatabasePrepare
   int      column,      // номер поля в запросе
   double&  value        // ссылка на переменную для получения значения
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

column

[in] Номер поля в запросе. Нумерация полей начинается с нуля и не может превышать значение DatabaseColumnsCount() - 1.

value

[out] Ссылка на переменную для записи значения поля.

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - индекс column больше значения DatabaseColumnsCount() -1.

Примечание

Значение можно получить только в том случае, если предварительно для запроса request был сделан хотя бы один вызов DatabaseRead().

Для чтения значения из следующей записи нужно предварительно вызвать DatabaseRead().

Смотри также

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnBlob

Получает из текущей записи значение поля в виде массива.

```
bool  DatabaseColumnBlob(
   int     request,      // хендл запроса, полученный в DatabasePrepare
   int     column,       // номер поля в запросе
   void&   data[]        // ссылка на переменную для получения значения
   );
```

Параметры

request

[in] Хендл запроса, полученный в DatabasePrepare().

column

[in] Номер поля в запросе. Нумерация полей начинается с нуля и не может превышать значение DatabaseColumnsCount() - 1.

data[]

[out] Ссылка на массив для записи значения поля.

Возвращаемое значение

Возвращает true в случае успеха или false, в случае ошибки. Для получения кода ошибки используйте GetLastError(), возможные ответы:

- ERR_DATABASE_INVALID_HANDLE (5121) - невалидный хендл запроса;
- ERR_DATABASE_NO_MORE_DATA (5126) - индекс column больше значения DatabaseColumnsCount() -1.

Примечание

Значение можно получить только в том случае, если предварительно для запроса request был сделан хотя бы один вызов DatabaseRead().

Для чтения значения из следующей записи нужно предварительно вызвать DatabaseRead().

Смотри также

DatabasePrepare, DatabaseColumnSize, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

Работа с DirectX

Функции и шейдеры DirectX 11 предназначены для 3D-визуализации прямо на ценовом графике.

Для создания трехмерной графики необходимо создать графический контекст (DXContextCreate) с нужным размером изображения. Далее подготавливаем вершинный и индексный буферы (DXBufferCreate), создаем вершинный и пиксельный шейдеры (DXShaderCreate). Этого достаточно чтобы вывести графику в цвете.

Для следующего уровня графики необходимо добавить входные параметры (DXInputSet), которые предназначены для передачи в шейдеры дополнительных параметров отрисовки. Это позволит задавать положение камеры и 3D-объекта, описать источники света, реализовать управление с помощью мышки и клавиатуры.

Таким образом, встроенные прямо в MQL5 функции позволяют создавать анимированные 3D-графики прямо в терминале MetaTrader 5 и не требуют для этого сторонних инструментов. Для работы функций видеокарта должна поддерживать DX 11 и шейдеры версии 5.0.

Для того чтобы начать работать с библиотекой, достаточно прочитать статью Как создать 3D-графику на DirectX в MetaTrader 5.

Функция	Действие
DXContextCreate	Создает графический контекст для отрисовки кадров указанного размера
DXContextSetSize	Изменяет размер кадра графического контекста, созданного в DXContextCreate()
DXContextSetSize	Получает размер кадра графического контекста, созданного в DXContextCreate()
DXContextClearColors	Устанавливает для буфера отрисовки все пиксели в указанный цвет
DXContextClearDepth	Очищает буфер глубины
DXContextGetColors	Получает из графического контекста изображение указанного размера и смещения
DXContextGetDepth	Получает буфер глубины отрисованного кадра
DXBufferCreate	Создает буфер указанного типа на основе массива данных
DXTextureCreate	Создает 2-мерную текстуру из прямоугольника заданного размера, вырезанного из переданного изображения
DXInputCreate	Создает входные параметры шейдера
DXInputSet	Устанавливает входные параметры шейдера
DXShaderCreate	Создает шейдер указанного типа
DXShaderSetLayout	Устанавливает разметку вершин для вершинного шейдера
DXShaderInputsSet	Устанавливает входные параметры шейдера

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
DXShaderTexturesSet	Устанавливает текстуры для шейдера
DXDraw	Отрисовывает вершины вершинного буфера, установленного в DXBufferSet()
DXDrawIndexed	Отрисовывает графические примитивы, описанные индексным буфером из DXBufferSet()
DXPrimiveTopologySet	Устанавливает тип примитивов для отрисовки с помощью DXDrawIndexed()
DXBufferSet	Устанавливает буфер для текущей отрисовки
DXShaderSet	Устанавливает шейдер для отрисовки
DXHandleType	Возвращает тип хендла
DXRelease	Освобождает хендл

© 2000-2025, MetaQuotes Ltd.

DXContextCreate

Создает графический контекст для отрисовки кадров указанного размера.

```
int  DXContextCreate(
   uint  width,       // ширина в пикселях
   uint  height       // высота в пикселях
   );
```

Параметры

width

[in] Ширина кадра в пикселях.

height

[in] Высота кадра в пикселях.

Возвращаемое значение

Хендл на созданный контекст или INVALID_HANDLE в случае ошибки. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Все графические объекты, созданные с помощью функций DXBufferCreate, DXInputCreate, DXShaderCreate и DXTextureCreate, можно использовать только в графическом контексте, в котором они были созданы.

Размер кадра впоследствии можно изменять в DXContextSetSize().

Созданный хендл, который больше не используется, должен быть явно освобожден с помощью функции DXRelease().

© 2000-2025, MetaQuotes Ltd.

DXContextSetSize

Изменяет размер кадра графического контекста, созданного в DXContextCreate().

```
bool  DXContextSetSize(
   int    context,       // хендл на графический контекст
   uint&  width,         // ширина в пикселях
   uint&  height         // высота в пикселях
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

width

[in] Ширина кадра в пикселях.

height

[in] Высота кадра в пикселях.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Изменять размер кадра графического контекста следует только между отрисовками кадров.

© 2000-2025, MetaQuotes Ltd.

DXContextGetSize

Получает размер кадра графического контекста, созданного в DXContextCreate().

```
bool  DXContextGetSize(
   int    context,       // хендл на графический контекст
   uint&  width,         // ширина в пикселях
   uint&  height         // высота в пикселях
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

width

[out] Ширина кадра в пикселях.

height

[out] Высота кадра в пикселях.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

DXContextClearColors

Устанавливает для буфера отрисовки все пиксели в указанный цвет.

```
bool  DXContextClearColors(
   int              context,       // хендл на графический контекст
   const DXVector&  color          // цвет
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

color

[in] Цвет для отрисовки.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Функция DXContextClearColors() может использоваться для очистки буфера цвета перед отрисовкой очередного кадра.

© 2000-2025, MetaQuotes Ltd.

DXContextClearDepth

Очищает буфер глубины.

```
bool  DXContextClearDepth(
   int  context      // хендл на графический контекст
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Функция DXContextClearDepth() может использоваться для очистки буфера глубины перед отрисовкой очередного кадра.

© 2000-2025, MetaQuotes Ltd.

DXContextGetColors

Получает из графического контекста изображение указанного размера и смещения.

```
bool  DXContextGetColors(
   int    context,                      // хендл на графический контекст
   uint&  image[],                      // массив пикселей изображения
   int    image_width=WHOLE_ARRAY,      // ширина изображения в пикселях
   int    image_height=WHOLE_ARRAY,     // высота изображения в пикселях
   int    image_offset_x=0,             // смещение по X
   int    image_offset_y=0              // смещение по Y
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

image

[out] Массив пикселей размера *image_width*image_height* в формате ARGB.

image_width=WHOLE_ARRAY

[in] Ширина изображения в пикселях.

image_height=WHOLE_ARRAY

[in] Высота изображения в пикселях.

image_offset_x=0

[in] Смещение по горизонтали.

image_offset_y=0

[in] Смещение по вертикали.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

DXContextGetDepth

Получает буфер глубины отрисованного кадра.

```
bool  DXContextGetDepth(
   int     context,      // хендл на графический контекст
   float&  image[]       // массив значений глубины
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

image

[out] Массив значений буфера глубины отрисованного кадра.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Возвращаемый буфер содержит в относительных единицах (от 0.0 до 1.0) глубину каждого пикселя отрисованного кадра, который может быть получен в DXContextGetColors().

© 2000-2025, MetaQuotes Ltd.

DXBufferCreate

Создает буфер указанного типа на основе массива данных.

```
int  DXBufferCreate(
   int                  context,              // хендл на графический контекст
   ENUM_DX_BUFFER_TYPE  buffer_type,          // тип создаваемого буфера
   const void&          data[],               // данные для буфера
   uint                 start=0,              // начальный индекс
   uint                 count=WHOLE_ARRAY     // количество элементов
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

buffer_type

[in] Тип буфера из перечисления ENUM_DX_BUFFER_TYPE.

data[]

[in] Данные для создания буфера.

start=0

[in] Индекс первого элемента массива, начиная с которого берутся значения из массива для создания буфера. По умолчанию берутся данные с начала массива.

count=WHOLE_ARRAY

[in] Количество значений. По умолчанию используется весь массив (count=WHOLE_ARRAY).

Возвращаемое значение

Хендл на созданный буфер или INVALID_HANDLE в случае ошибки. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Для индексного буфера массив *data[]* должен иметь тип uint, а для вершинного буфера передается массив структур, описывающий вершины.

Созданный хендл, который больше не используется, должен быть явно освобожден с помощью функции DXRelease().

ENUM_DX_BUFFER_TYPE

Идентификатор	Значение	Описание
DX_BUFFER_VERTEX	1	Вершинный буфер
DX_BUFFER_INDEX	2	Индексный буфер

© 2000-2025, MetaQuotes Ltd.

DXTextureCreate

Создает 2-мерную текстуру из прямоугольника заданного размера, вырезанного из переданного изображения.

```
int  DXTextureCreate(
   int             context,          // хендл на графический контекст
   ENUM_DX_FORMAT  format,           // формат цвета пикселя
   uint            width,            // ширина исходного изображения
   uint            height,           // высота исходного изображения
   const void&     data[],           // массив пикселей исходного изображения
   uint            data_x,           // X-координата прямоугольника для создания тексту
   uint            data_y,           // Y-координата прямоугольника для создания тексту
   uint            data_width,       // ширина прямоугольника для создания текстуры
   uint            data_height       // высота прямоугольника для создания текстуры
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

format

[in] Формат цвета пикселя, задаваемый из перечисления ENUM_DX_FORMAT.

width

[in] Ширина изображения, на основе которого создается текстура.

height

[in] Высота изображения, на основе которого создается текстура.

data

[in] Массив пикселей изображения, на основе которого создается текстура.

data_x

[in] X-координата прямоугольника (смещение по горизонтали), из которого будет создана текстура.

data_y

[in] Y-координата прямоугольника (смещение по вертикали), из которого будет создана текстура.

data_width

[in] Ширина прямоугольника, из которого будет создана текстура.

data_height

[in] Высота прямоугольника, из которого будет создана текстура.

Возвращаемое значение

Хендл на текстуру или INVALID_HANDLE в случае ошибки. Для получения кода ошибки нужно вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

Примечание

Созданный хендл, который больше не используется, должен быть явно освобожден с помощью функции DXRelease().

ENUM_DX_FORMAT

Идентификатор	Значение	Соответствие в 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

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение	Соответствие в DXGI_FORMAT
DX_FORMAT_X32_TYPELESS_G8X24_UINT	22	DXGI_FORMAT_X32_TYPELESS_G8X24_UINT
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

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение	Соответствие в DXGI_FORMAT
DX_FORMAT_R8G8_UINT	50	DXGI_FORMAT_R8G8_UINT
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

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение	Соответствие в DXGI_FORMAT
DX_FORMAT_BC4_SNORM	81	DXGI_FORMAT_BC4_SNORM
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

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Значение	Соответствие в DXGI_FORMAT
DX_FORMAT_NV11	110	DXGI_FORMAT_NV11
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

© 2000-2025, MetaQuotes Ltd.

DXInputCreate

Создает входные параметры шейдера.

```
int  DXInputCreate(
   int   context,        // хендл на графический контекст
   uint  input_size      // размер входных параметров в байтах
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

input_size

[in] Размер структуры параметров в байтах.

Возвращаемое значение

Хендл на входные параметры шейдера или INVALID_HANDLE в случае ошибки. Для получения кода ошибки нужно вызвать функцию GetLastError().

Созданный хендл, который больше не используется, должен быть явно освобожден с помощью функции DXRelease().

© 2000-2025, MetaQuotes Ltd.

DXInputSet

Устанавливает входные параметры шейдера.

```
bool  DXInputSet(
   int           input,       // хендл на графический контекст
   const void&   data         // данные для установки
   );
```

Параметры

input

[in] Хендл входных параметров шейдера, полученного в DXInputCreate().

data

[in] Данные для установки входных параметров шейдера.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

DXShaderCreate

Создает шейдер указанного типа.

```
int  DXShaderCreate(
   int                  context,          // хендл на графический контекст
   ENUM_DX_SHADER_TYPE  shader_type,      // тип шейдера
   const string         source,           // исходный код шейдера
   const string         entry_point,      // точка входа
   string&              compile_error     // строка для получения сообщений компилятор
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

shader_type

[out] Значение из перечисления ENUM_DX_SHADER_TYPE.

source

[in] Исходный код шейдера, написанный на HLSL 5.

entry_point

[in] Точка входа - название функции в исходном коде.

compile_error

[in] Строка для получения ошибок компиляции.

Возвращаемое значение

Хендл на шейдер или INVALID_HANDLE в случае ошибки. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Созданный хендл, который больше не используется, должен быть явно освобожден с помощью функции DXRelease().

ENUM_DX_SHADER_TYPE

Идентификатор	Значение	Описание
DX_SHADER_VERTEX	0	Вершинный шейдер
DX_SHADER_GEOMETRY	1	Геометрический шейдер
DX_SHADER_PIXEL	2	Пиксельный шейдер

© 2000-2025, MetaQuotes Ltd.

DXShaderSetLayout

Устанавливает разметку вершин для вершинного шейдера.

```
bool  DXShaderSetLayout(
   int                     shader,        // хендл шейдера
   const DXVertexLayout&   layout[]       //
   );
```

Параметры

shader

[in] Хендл вершинного шейдера, созданного в DXShaderCreate().

layout[]

[in] Массив описания полей вершин. Описание задается структурой 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.
  };
```

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Разметка должна соответствовать типу вершин в установленном вершинном буфере и входному типу вершин, используемых в точке входа в коде вершинного шейдера.

Вершинный буфер для шейдера устанавливается в DXBufferSet().

Структура DXVertexLayout является версий структуры D3D11_INPUT_ELEMENT_DESC из MSDN.

© 2000-2025, MetaQuotes Ltd.

DXShaderInputsSet

Устанавливает входные параметры шейдера.

```
bool  DXShaderInputsSet(
   int          shader,        // хендл шейдера
   const int&   inputs[]       // массив хендлов входных параметров
   );
```

Параметры

shader

[in] Хендл шейдера, созданного в DXShaderCreate().

inputs[]

[in] Массив хендлов входных параметров, созданных с помощью DXInputCreate().

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Размер массива входных параметров должен быть равен количеству объектов cbuffer, объявленных в коде шейдера.

© 2000-2025, MetaQuotes Ltd.

DXShaderTexturesSet

Устанавливает текстуры для шейдера.

```
bool  DXShaderTexturesSet(
   int           shader,          // хендл шейдера
   const  int&   textures[]       // массив хендлов структур
   );
```

Параметры

shader

[in] Хендл шейдера, созданного в DXShaderCreate().

textures[]

[in] Массив хендлов текстур, созданных с помощью DXTextureCreate().

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Размер массива текстур должен быть равен количеству объектов Texture2D, объявленных в коде шейдера.

© 2000-2025, MetaQuotes Ltd.

DXDraw

Отрисовывает вершины вершинного буфера, установленного в DXBufferSet().

```
bool  DXDraw(
   int   context,               // хендл на графический контекст
   uint  start=0,               // индекс первой вершины
   uint  count=WHOLE_ARRAY      // количество вершин
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

start=0

[in] Индекс первой вершины для отрисовки.

count=WHOLE_ARRAY

[in] Количество вершин для отрисовки.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Для отрисовки вершин предварительно должны быть установлены шейдеры с помощью DXShaderSet().

© 2000-2025, MetaQuotes Ltd.

DXDrawIndexed

Отрисовывает графические примитивы, описанные индексным буфером из DXBufferSet().

```
bool  DXDrawIndexed(
   int   context,               // хендл на графический контекст
   uint  start=0,               // индекс первого примитива
   uint  count=WHOLE_ARRAY      // количество примитивов
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

start=0

[in] Индекс первого примитива для отрисовки.

count=WHOLE_ARRAY

[in] Количество примитивов для отрисовки.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Тип примитивов, описываемых индексным буфером задается с помощью DXPrimiveTopologySet().

Для отрисовки примитивов предварительно должен быть установлен вершинный буфер в DXBufferSet().

Для отрисовки примитивов предварительно должны быть установлены шейдеры с помощью DXShaderSet().

© 2000-2025, MetaQuotes Ltd.

DXPrimiveTopologySet

Устанавливает тип примитивов для отрисовки с помощью DXDrawIndexed().

```
bool  DXPrimiveTopologySet(
   int                         context,              // хендл на графический контекс
   ENUM_DX_PRIMITIVE_TOPOLOGY  primitive_topology    // тип примитива
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

primitive_topology

[in] Значение из перечисления ENUM_DX_PRIMITIVE_TOPOLOGY.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

ENUM_DX_PRIMITIVE_TOPOLOGY

Идентификатор	Значение	Соответствие в 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

© 2000-2025, MetaQuotes Ltd.

DXBufferSet

Устанавливает буфер для текущей отрисовки.

```
bool  DXBufferSet(
   int  context,                // хендл на графический контекст
   int  buffer,                 // хендл вершинного или индексного буфера
   uint  start=0,               // начальный индекс
   uint  count=WHOLE_ARRAY      // количество элементов
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

buffer

[in] Хендл вершинного или индексного буфера, созданного в DXBufferCreate().

start=0

[in] Индекс первого элемента буфера. По умолчанию используются данные с начала буфера.

count=WHOLE_ARRAY

[in] Количество значений, которые нужно использовать. По умолчанию все значения буфера.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Функцию DXBufferSet() необходимо вызывать для установки вершинного и индексного буферов для отрисовки с помощью DXDraw().

© 2000-2025, MetaQuotes Ltd.

DXShaderSet

Устанавливает шейдер для отрисовки.

```
bool  DXShaderSet(
   int  context,      // хендл на графический контекст
   int  shader        // хендл шейдера
   );
```

Параметры

context

[in] Хендл на графический контекст, созданный в DXContextCreate().

shader

[in] Хендл шейдера, созданного в DXShaderCreate().

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Для отрисовки одновременно может использоваться несколько типов шейдеров - вершинный, геометрический, пиксельный.

© 2000-2025, MetaQuotes Ltd.

DXHandleType

Возвращает тип хендла.

```
ENUM_DX_HANDLE_TYPE  DXHandleType(
   int  handle      // хендл
   );
```

Параметры

handle

[in] Хендл.

Возвращаемое значение

Значение из перечисления ENUM_DX_HANDLE_TYPE

ENUM_DX_HANDLE_TYPE

Идентификатор	Значение	Описание
DX_HANDLE_INVALID	0	Невалидный хендл
DX_HANDLE_CONTEXT	1	Хендл на графический контекст
DX_HANDLE_SHADER	2	Хендл на шейдер
DX_HANDLE_BUFFER	3	Хендл на вершинный или индексный буфер
DX_HANDLE_INPUT	4	Хендл на входные параметры шейдера
DX_HANDLE_TEXTURE	5	Хендл на текстуру

© 2000-2025, MetaQuotes Ltd.

DXRelease

Освобождает хендл.

```
bool  DXRelease(
   int  handle      // хендл
   );
```

Параметры

context

[in] Освобождаемый хендл.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки нужно вызвать функцию GetLastError().

Примечание

Все созданные хендлы, которые больше не используются, должны быть явно освобождены с помощью функции DXRelease().

© 2000-2025, MetaQuotes Ltd.

Модуль MetaTrader для интеграции с Python

MQL5 предназначен для разработки высокопроизводительных торговых приложений на финансовых рынках и не имеет аналогов среди других специализованных языков, используемых в алготрейдинге. Синтаксис и скорость работы программ на MQL5 максимально приближены к C++, имеется поддержка OpenCL и интеграция с MS Visual Studio, доступны библиотеки статистики, нечеткой логики и ALGLIB. В среде разработки MetaEditor также имеется нативная поддержка .NET библиотек с "умным" импортом функций без написания специальных оберток, можно использовать сторонние C++ DLL. Файлы исходных кодов на C++ (CPP и H) можно редактировать и компилировать в DLL прямо из редактора — для этого используется Microsoft Visual Studio, установленный на компьютере пользователя.

Python является современным высокоуровневым языком программирования для разработки сценариев и приложений. Содержит множество библиотек для машинного обучения, автоматизации процессов, анализа и визуализации данных.

Пакет MetaTrader для Python предназначен для удобного и быстрого получения биржевой информации через межпроцессное взаимодействие прямо из терминала MetaTrader 5. Полученные таким образом данные можно дальше использовать для статистических вычислений и машинного обучения.

Установка пакета в командной строке:

```
pip install MetaTrader5
```

Обновление пакета в командной строке:

```
pip install --upgrade MetaTrader5
```

Функции для интеграции MetaTrader 5 и Python

Функция	Действие
initialize	Устанавливает соединение с терминалом MetaTrader 5
login	Подключается к торговому счету с указанными параметрами
shutdown	Закрывает ранее установленное подключение к терминалу MetaTrader 5
version	Возвращает версию терминала MetaTrader 5
last_error	Возвращает информацию о последней ошибке
account_info	Получает информацию о текущем торговом счете
terminal_Info	Получает состояние и параметры подключенного терминала MetaTrader 5
symbols_total	Получает количество всех финансовых инструментов в терминале MetaTrader 5
symbols_get	Получает все финансовые инструменты из терминала MetaTrader 5
symbol_info	Получает информацию по указанному финансовому инструменту

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
symbol_info_tick	Получает последний тик по указанному финансовому инструменту
symbol_select	Выбирает символ в окне MarketWatch или убирает символ из этого окна
market_book_add	Производит подписку терминала MetaTrader 5 на получение событий об изменениях в стакане по указанному символу
market_book_get	Возвращает кортеж из BookInfo, содержащий записи стакана цен указанного символа
market_book_release	Отменяет подписку терминала MetaTrader 5 на получение событий об изменениях в стакане по указанному символу
copy_rates_from	Получает бары из терминала MetaTrader 5, начиная с указанной даты
copy_rates_from_pos	Получает бары из терминала MetaTrader 5, начиная с указанного индекса
copyrates_range	Получает бары в указанном диапазоне дат из терминала MetaTrader 5
copy_ticks_from	Получает тики из терминала MetaTrader 5, начиная с указанной даты
copy_ticks_range	Получает тики за указанный диапазон дат из терминала MetaTrader 5
orders_total	Получает количество действующих ордеров.
orders_get	Получает действующие ордеры с возможность фильтрации по символу или тикету
order_calc_margin	Возвращает размер маржи в валюте счета для проведения указанной торговой операции
order_calc_profit	Возвращает размер прибыли в валюте счета для указанной торговой операции
order_check	Проверяет достаточность средств для совершения требуемой торговой операции
order_send	Отправляет из терминала на торговый сервер запрос на совершение торговой операции.
positions_total	Получает количество открытых позиций
positions_get	Получает открытые позиции с возможностью фильтрации по символу или тикету
history_orders_total	Получает количество ордеров в торговой истории в указанном интервале
history_orders_get	Получает ордеры из торговой истории с возможностью фильтрации по тикету или позиции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие
history_deals_total	Получает количество сделок в торговой истории в указанном интервале
history_deals_get	Получает сделки из торговой истории с возможностью фильтрации по тикету или позиции

Пример подключения Python к MetaTrader 5

1. Скачайте последнюю версию Python 3.8 со страницы https://www.python.org/downloads/windows
2. При установке Python отметьте чек-бокс "Add Python 3.8 to PATH%", чтобы можно было из командной строки запускать скрипты на Python.
3. Установите модуль MetaTrader5 из командной строки

```
pip install MetaTrader5
```

4. Добавьте пакеты matplotlib и pandas

```
pip install matplotlib
pip install pandas
```

5. Запустите тестовый скрипт

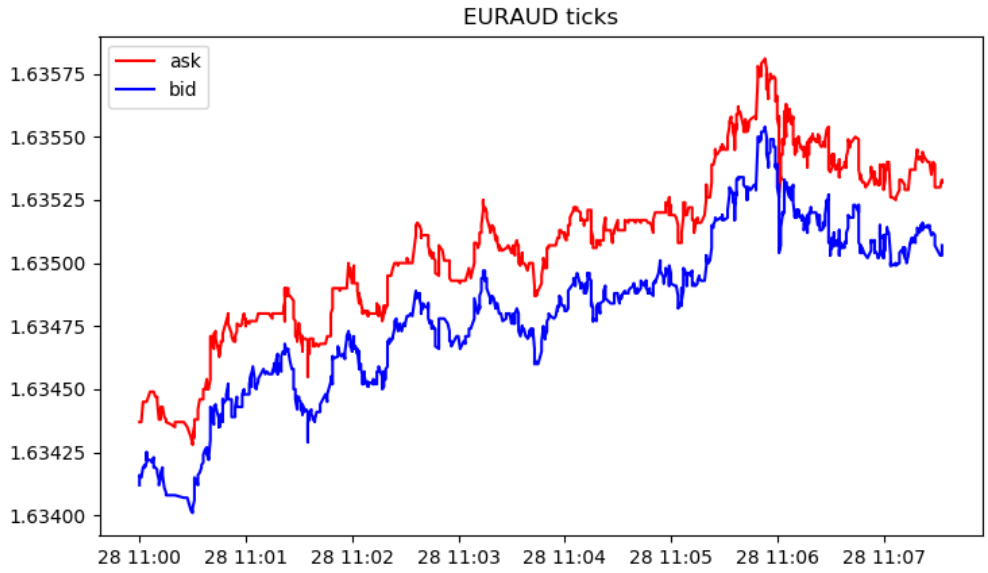
```
from datetime import datetime
import matplotlib.pyplot as plt
import pandas as pd
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import MetaTrader5 as mt5

# подключимся к MetaTrader 5
if not mt5.initialize():
    print("initialize() failed")
    mt5.shutdown()

# запросим статус и параметры подключения
print(mt5.terminal_info())
# получим информацию о версии MetaTrader 5
print(mt5.version())

# запросим 1000 тиков с EURAUD
euraud_ticks = mt5.copy_ticks_from("EURAUD", datetime(2020,1,28,13), 1000, mt5.COPY_TI
# запросим тики с AUDUSD в интервале 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,

# получим бары с разных инструментов разными способами
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)
```

© 2000-2025, MetaQuotes Ltd.

```
# завершим подключение к MetaTrader 5
mt5.shutdown()

#DATA
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
# создадим из полученных данных DataFrame
ticks_frame = pd.DataFrame(euraud_ticks)
# сделаем отрисовку тиков на графике
plt.plot(ticks_frame['time'], ticks_frame['ask'], 'r-', label='ask')
plt.plot(ticks_frame['time'], ticks_frame['bid'], 'b-', label='bid')

# выведем легенды
plt.legend(loc='upper left')

# добавим заголовок
plt.title('EURAUD ticks')

# покажем график
plt.show()
```

© 2000-2025, MetaQuotes Ltd.

6. Получите данные и график




```
[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)
```

© 2000-2025, MetaQuotes Ltd.

```
(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)
(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)
```

© 2000-2025, MetaQuotes Ltd.

initialize

Устанавливает соединение с терминалом MetaTrader 5. Существует 3 варианта вызова.

Вызов без параметров. Терминал для подключения будет найден автоматически.

```
initialize()
```

Вызов с указанием пути к терминалу MetaTrader 5, к которому необходимо подключиться.

```
initialize(
   path                       // путь к EXE-файлу терминала MetaTrader 5
   )
```

Вызов с указанием пути и параметров торгового счета.

```
initialize(
   path,                      // путь к EXE-файлу терминала MetaTrader 5
   login=LOGIN,               // номер счета
   password="PASSWORD",       // пароль
   server="SERVER",           // имя сервера, как оно задано в терминале
   timeout=TIMEOUT,           // таймаут
   portable=False             // режим portable
   )
```

Параметры

path

[in] Путь к файлу metatrader.exe или metatrader64.exe. Необязательный неименованный параметр. Указывается первым без имени параметра. Если путь не указан, модуль попытается найти исполняемый файл самостоятельно.

login=LOGIN

[in] Номер торгового счета. Необязательный именованный параметр. Если не указан, то будет использоваться последний торговый счет.

password="PASSWORD"

[in] Пароль к торговому счету. Необязательный именованный параметр. Если пароль не указан, то автоматически подставляется сохраненный в базе терминала пароль для указанного торгового счета.

server="SERVER"

[in] Имя торгового сервера. Необязательный именованный параметр. Если сервер не указан, то автоматически подставляется сохраненный в базе терминала сервер для указанного торгового счета.

timeout=TIMEOUT

[in] Таймаут в миллисекундах, который дается на подключение. Необязательный именованный параметр. Если не указан, то будет использоваться значение 60000 (60 секунд).

portable=False

[in] Признак запуска терминала в режиме portable. Необязательный именованный параметр. Если не указан, то будет использоваться значение False.

© 2000-2025, MetaQuotes Ltd.

Возвращаемое значение

Возвращает True в случае успешного подключения к терминалу MetaTrader 5, иначе False.

Примечание

Если потребуется, при выполнении вызова initialize() терминал MetaTrader 5 будет запущен для выполнения соединения.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5 на указанный торговый счет
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# выведем информацию о состоянии подключения, названии сервера и торговом счете
print(mt5.terminal_info())
# выведем информацию о версии MetaTrader 5
print(mt5.version())

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

Смотри также

shutdown, terminal_info, version

© 2000-2025, MetaQuotes Ltd.

login

Подключается к торговому счету с указанными параметрами.

```
login(
   login,                    // номер счета
   password="PASSWORD",      // пароль
   server="SERVER",          // имя сервера, как оно задано в терминале
   timeout=TIMEOUT           // таймаут
   )
```

Параметры

login

[in] Номер торгового счета. Обязательный неименованный параметр.

password

[in] Пароль к торговому счету. Необязательный именованный параметр. Если пароль не указан, то автоматически подставляется сохраненный в базе терминала пароль.

server

[in] Имя торгового сервера. Необязательный именованный параметр. Если сервер не указан, то автоматически подставляется последний использованный сервер.

timeout=TIMEOUT

[in] Таймаут в миллисекундах, который дается на подключение. Необязательный именованный параметр. Если не указан, то будет использоваться значение 60000 (60 секунд). Если подключение не установится за это время, то вызов будет принудительно завершен и сгенерируется исключение.

Возвращаемое значение

True в случае успешного подключения к торговому счету, иначе False.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# выведем информацию о версии MetaTrader 5
print(mt5.version())
# подключимся к торговому счету без указания пароля и сервера
account=17221085
authorized=mt5.login(account)  # пароль будет взят из базы терминала, если указано пом
if authorized:
    print("connected to account #{}".format(account))
```

© 2000-2025, MetaQuotes Ltd.

```
else:
    print("failed to connect at account #{}, error code: {}".format(account, mt5.last_

# теперь подключимся к другому торговому счету с указанием пароля
account=25115284
authorized=mt5.login(account, password="gqrtz0lbdm")
if authorized:
    # выведем данные о торговом счете как есть
    print(mt5.account_info())
    # выведем данные о торговом счете в виде списка
    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_

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

```
Результат:
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
```

© 2000-2025, MetaQuotes Ltd.

```
margin_so_so=30.0
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.
```

Смотри также

initialize, shutdown

© 2000-2025, MetaQuotes Ltd.

shutdown

Закрывает ранее установленное подключение к терминалу MetaTrader 5.

```
shutdown()
```

Возвращаемое значение

Нет.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed")
    quit()

# выведем информацию о состоянии подключения, названии сервера и торговом счете
print(mt5.terminal_info())
# выведем информацию о версии MetaTrader 5
print(mt5.version())

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

Смотри также

initialize, login_py, terminal_info, version

© 2000-2025, MetaQuotes Ltd.

version

Возвращает версию терминала MetaTrader 5.

```
version()
```

Возвращаемое значение

Возвращает номер версии, номер билда и дату релиза терминала MetaTrader 5. В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Функция version() возвращает номер версии, номер билда и дату релиза терминала в виде кортежа из трех значений:

Тип	Описание	Пример значения
integer	Версия терминала MetaTrader 5	500
integer	Номер билда	2007
string	Дата релиза билда	'25 Feb 2019'

Пример:

```
import MetaTrader5 as mt5
import pandas as pd
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# выведем информацию о версии терминала MetaTrader 5
print(mt5.version())

# выведем информацию о состоянии подключения, названии сервера и торговом счете как ес
print(mt5.terminal_info())
print()

# получим свойства в виде словаря
terminal_info_dict=mt5.terminal_info()._asdict()
# преобразуем словарь в DataFrame и выведем на печать
df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
print("terminal_info() as dataframe:")
print(df[:-1])
```

© 2000-2025, MetaQuotes Ltd.

```
# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()

Результат:
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
```

Смотри также

initialize, shutdown, terminal_info

© 2000-2025, MetaQuotes Ltd.

last_error

Возвращает информацию о последней ошибке.

```
last_error()
```

Возвращаемое значение

Возвращает код последней ошибки и её описание в виде кортежа.

Примечание

last_error() позволяет получить код ошибки в случае неуспешного выполнения какой-либо функции библиотеки MetaTrader 5. Является аналогом GetLastError(), но используются свои коды ошибок. Возможные значения:

Константа	Значение	Описание
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

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

Смотри также

version, GetLastError

© 2000-2025, MetaQuotes Ltd.

account_info

Получает информацию о текущем торговом счете.

```
account_info()
```

Возвращаемое значение

Возвращает информацию в виде структуры именованных кортежей (namedtuple). В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Функция возвращает за один вызов всю информацию, которую можно получить с помощью AccountInfoInteger, AccountInfoDouble и AccountInfoString.

Пример:

```
import MetaTrader5 as mt5
import pandas as pd
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# подключимся к торговому счету с указанием пароля и сервера
authorized=mt5.login(25115284, password="gqz0343lbdm")
if authorized:
    account_info=mt5.account_info()
    if account_info!=None:
        # выведем данные о торговом счете как есть
        print(account_info)
        # выведем данные о торговом счете в виде словаря
        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()

        # преобразуем словарь в DataFrame и выведем на печать
        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

# завершим подключение к терминалу MetaTrader 5
```

© 2000-2025, MetaQuotes Ltd.

```
mt5.shutdown()


Результат:
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
```

© 2000-2025, MetaQuotes Ltd.

```
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.
```

Смотри также

initialize, shutdown, login

© 2000-2025, MetaQuotes Ltd.

terminal_info

Получает состояние и настройки подключенного клиентского терминала MetaTrader 5.

```
terminal_info()
```

Возвращаемое значение

Возвращает информацию в виде структуры именованных кортежей (namedtuple). В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Функция возвращает за один вызов всю информацию, которую можно получить с помощью TerminalInfoInteger, TerminalInfoDouble и TerminalInfoDouble.

Пример:

```
import MetaTrader5 as mt5
import pandas as pd
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# выведем информацию о версии MetaTrader 5
print(mt5.version())
# выведем информацию о настройках и состоянии терминала
terminal_info=mt5.terminal_info()
if terminal_info!=None:
    # выведем данные о терминале как есть
    print(terminal_info)
    # выведем данные в виде списка
    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()
    # преобразуем словарь в DataFrame и выведем на печать
    df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
    print("terminal_info() as dataframe:")
    print(df)

# завершим подключение к терминалу MetaTrader 5
```

© 2000-2025, MetaQuotes Ltd.

```
mt5.shutdown()


Результат:
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
```

© 2000-2025, MetaQuotes Ltd.

```
15         retransmission                   0.063593
16                company  MetaQuotes Software Corp.
17                   name                MetaTrader 5
18               language                     Russian
```

Смотри также

initialize, shutdown, version

© 2000-2025, MetaQuotes Ltd.

symbols_total

Получает количество всех финансовых инструментов в терминале MetaTrader 5.

```
symbols_total()
```

Возвращаемое значение

Целочисленное значение.

Примечание

Функция является аналогом SymbolsTotal(), но при этом возвращает количество всех символов, включая пользовательские и отключенные в MarketWatch.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# получим количество финансовых инструментов
symbols=mt5.symbols_total()
if symbols>0:
    print("Total symbols =",symbols)
else:
    print("Symbols not found")

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

Смотри также

symbols_get, symbol_select, symbol_info

© 2000-2025, MetaQuotes Ltd.

symbols_get

Получает все финансовые инструменты из терминала MetaTrader 5.

```
symbols_get(
   group="GROUP"       // фильтр для отбора символов
)
```

group="GROUP"

[in] Фильтр для отбора группы только необходимых символов. Необязательный параметр. Если группа задана, то функция вернет только те символы, которые удовлетворяют заданному критерию.

Возвращаемое значение

Возвращает символы в виде кортежа (tuple). В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Параметр *group* позволяет отфильтровать символы по имени, допустимо использовать '*' в начале и конце строки.

Параметр *group* можно использовать как в качестве именованного, так и в качестве неименованного параметра. Оба варианта работают одинаково. Использование именованного варианта (group="GROUP") делает код более понятным.

Параметр *group* может содержать несколько условий, разделенных запятыми. Условие можно задавать как маску с использованием '*'. Для исключения можно использовать символ логического отрицания '!'. При этом все условия применяются последовательно, то есть сначала необходимо указать условия включения в группу, а затем условие исключения. Например, *group*="*, !EUR" означает, что сначала нужно выбрать все символы и затем исключить те из них , что содержат в имени "EUR".

В отличие от symbol_info() функция symbols_get() возвращает за один вызов информацию по всем запрошенным символам.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# получим все символы
symbols=mt5.symbols_get()
print('Symbols: ', len(symbols))
count=0
```

© 2000-2025, MetaQuotes Ltd.

```
# выведем 5 первых
for s in symbols:
    count+=1
    print("{}. {}".format(count,s.name))
    if count==5: break
print()

# получим символы, чьи имена содержат RU
ru_symbols=mt5.symbols_get("*RU*")
print('len(*RU*): ', len(ru_symbols))
for s in ru_symbols:
    print(s.name)
print()

# получим символы, имена которых не содержат USD,EUR,JPY и 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)

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()

Результат:
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_c
AUDNZD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CADCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDCAD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
```

© 2000-2025, 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
```

Смотри также

symbols_total, symbol_select, symbol_info

© 2000-2025, MetaQuotes Ltd.

symbol_info

Получает информацию по указанному финансовому инструменту.

```
symbol_info(
   symbol      // имя финансового инструмента
)
```

symbol

[in] Имя финансового инструмента. Обязательный неименованный параметр.

Возвращаемое значение

Возвращает информацию в виде структуры именованных кортежей (namedtuple). В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Функция возвращает за один вызов всю информацию, которую можно получить с помощью SymbolInfoInteger, SymbolInfoDouble и SymbolInfoString.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# попробуем включить показ символа EURJPY в MarketWatch
selected=mt5.symbol_select("EURJPY",True)
if not selected:
    print("Failed to select EURJPY")
    mt5.shutdown()
    quit()

# выведем свойства по символу EURJPY
symbol_info=mt5.symbol_info("EURJPY")
if symbol_info!=None:
    # выведем данные о терминале как есть
    print(symbol_info)
    print("EURJPY: spread =",symbol_info.spread,"  digits =",symbol_info.digits)
    # выведем свойства символа в виде списка
    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]))
```

© 2000-2025, MetaQuotes Ltd.

```
# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()


Результат:
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
```

© 2000-2025, 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
```

© 2000-2025, 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
```

Смотри также

account_info, terminal_info

© 2000-2025, MetaQuotes Ltd.

symbol_info_tick

Получает последний тик по указанному финансовому инструменту.

```
symbol_info_tick(
   symbol      // имя финансового инструмента
)
```

symbol

[in] Имя финансового инструмента. Обязательный неименованный параметр.

Возвращаемое значение

Возвращает информацию в виде кортежа (tuple). В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Функция является аналогом SymbolInfoTick.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# попробуем включить показ символа GBPUSD в MarketWatch
selected=mt5.symbol_select("GBPUSD",True)
if not selected:
    print("Failed to select GBPUSD")
    mt5.shutdown()
    quit()

# выведем последний тик по символу GBPUSD
lasttick=mt5.symbol_info_tick("GBPUSD")
print(lasttick)
# выведем значения полей тика в виде списка
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]))

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

```
Результат:
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
```

Смотри также

symbol_info, symbol_info

© 2000-2025, MetaQuotes Ltd.

symbol_select

Выбирает символ в окне MarketWatch или убирает символ из этого окна.

```
symbol_select(
   symbol,       // имя финансового инструмента
   enable=None   // включить или выключить
)
```

symbol

[in] Имя финансового инструмента. Обязательный неименованный параметр.

enable

[in] Переключатель. Необязательный неименованный параметр. Если значение false, то символ должен быть убран из окна MarketWatch, в противном случае символ должен быть выбран в окне MarketWatch. Символ не может быть убран, если есть открытые графики с этим символом или есть открытые позиции по этому символу.

Возвращаемое значение

True в случае успешного выполнения, иначе - False.

Примечание

Функция является аналогом SymbolSelect.

Пример:

```
import MetaTrader5 as mt5
import pandas as pd
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# установим подключение к терминалу MetaTrader 5
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# попробуем включить показ символа EURCAD в 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()

    # получим свойства символа в виде словаря
    print("Show symbol_info()._asdict():")
    symbol_info_dict = symbol_info._asdict()
```

© 2000-2025, MetaQuotes Ltd.

```
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
    print()

    # преобразуем словарь в DataFrame и выведем на печать
    df=pd.DataFrame(list(symbol_info_dict.items()),columns=['property','value'])
    print("symbol_info_dict() as dataframe:")
    print(df)

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()


Результат:
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
```

© 2000-2025, 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
```

© 2000-2025, 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]
```

Смотри также

symbol_info

© 2000-2025, MetaQuotes Ltd.

market_book_add

Производит подписку терминала MetaTrader 5 на получение событий об изменениях в стакане по указанному символу.

```
market_book_add(
   symbol      // имя финансового инструмента
)
```

symbol

[in] Имя финансового инструмента. Обязательный неименованный параметр.

Возвращаемое значение

True в случае успешного выполнения, иначе - False.

Примечание

Функция является аналогом MarketBookAdd.

Смотри также

market_book_get, market_book_release, Структура стакана цен

© 2000-2025, MetaQuotes Ltd.

market_book_get

Возвращает кортеж из BookInfo, содержащий записи стакана цен указанного символа.

```
market_book_get(
   symbol      // имя финансового инструмента
)
```

symbol

[in] Имя финансового инструмента. Обязательный неименованный параметр.

Возвращаемое значение

Возвращает содержимое стакана в виде кортежа из записей BookInfo, содержащих тип заявки, цену и объем в лотах. BookInfo является аналогом структуры MqlBookInfo.

В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Предварительно с помощью функции market_book_add() должна быть оформлена подписка на получение событий изменения в стакане цен.

Функция является аналогом MarketBookGet.

Пример:

```
import MetaTrader5 as mt5
import time
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print("")

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    # завершим подключение к терминалу MetaTrader 5
    mt5.shutdown()
    quit()

# подпишемся на получение обновлений в стакане по символу EURUSD (Depth of Market)
if mt5.market_book_add('EURUSD'):
   # получим 10 раз в цикле данные из стакана
   for i in range(10):
        # получим содержимое стакана (Depth of Market)
        items = mt5.market_book_get('EURUSD')
        # выведем весь стакан одной строкой как есть
        print(items)
        # теперь выведем каждую заявку отдельно для наглядности
```

© 2000-2025, MetaQuotes Ltd.

```
        if items:
            for it in items:
                # содержимое заявки
                print(it._asdict())
        # сделаем паузу в 5 секунд перед следующим запросом данных из стакана
        time.sleep(5)
   # отменим подписку на обновления в стакане (Depth of Market)
   mt5.market_book_release('EURUSD')
else:
    print("mt5.market_book_add('EURUSD') failed, error code =",mt5.last_error())

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

Результат:

```
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}
```

© 2000-2025, 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}
```

Смотри также

market_book_add, market_book_release, Структура стакана цен

© 2000-2025, MetaQuotes Ltd.

market_book_release

Отменяет подписку терминала MetaTrader 5 на получение событий об изменениях в стакане по указанному символу.

```
market_book_release(
   symbol      // имя финансового инструмента
)
```

symbol

[in] Имя финансового инструмента. Обязательный неименованный параметр.

Возвращаемое значение

True в случае успешного выполнения, иначе - False.

Примечание

Функция является аналогом MarketBookRelease.

Смотри также

market_book_add, market_book_get, Структура стакана цен

© 2000-2025, MetaQuotes Ltd.

copy_rates_from

Получает бары из терминала MetaTrader 5, начиная с указанной даты.

```
copy_rates_from(
   symbol,       // имя символа
   timeframe,    // таймфрейм
   date_from,    // дата открытия начального бара
   count         // количество баров
   )
```

Параметры

symbol

[in] Имя финансового инструмента, например, "EURUSD". Обязательный неименованный параметр.

timeframe

[in] Таймфрейм, для которого запрашиваются бары. Задается значением из перечисления TIMEFRAME. Обязательный неименованный параметр.

date_from

[in] Дата открытия первого бара из запрашиваемой выборки. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр.

count

[in] Количество баров, которое необходимо получить. Обязательный неименованный параметр.

Возвращаемое значение

Возвращает бары в виде массива numpy с именованными столбцами time, open, high, low, close, tick_volume, spread и real_volume. В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Для дополнительной информации смотрите функцию CopyRates().

Возвращаются только данные, дата которых меньше (раньше) или равна указанной. При этом интервал задается и учитывается с точностью до секунды. То есть дата открытия любого бара всегда равна или меньше указанной.

Терминал MetaTrader 5 отдает бары только в пределах истории, доступной пользователю на графиках. Количество баров, которые доступны пользователю, задается в настройках параметром "Макс. баров в окне".

Python при создании объекта datetime использует таймзону локального времени, в то время как терминал MetaTrader 5 хранит время тиков и открытия баров в UTC таймзоне (без смещения). Поэтому, для выполнения функций, использующих время, необходимо создавать datetime в UTC-времени. Данные, полученные из терминала MetaTrader 5, имеют UTC-время.

TIMEFRAME является перечислением с возможными значениями периодов графика

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
TIMEFRAME_M1	1 минута
TIMEFRAME_M2	2 минуты
TIMEFRAME_M3	3 минуты
TIMEFRAME_M4	4 минуты
TIMEFRAME_M5	5 минут
TIMEFRAME_M6	6 минут
TIMEFRAME_M10	10 минут
TIMEFRAME_M12	12 минут
TIMEFRAME_M12	15 минут
TIMEFRAME_M20	20 минут
TIMEFRAME_M30	30 минут
TIMEFRAME_H1	1 час
TIMEFRAME_H2	2 часа
TIMEFRAME_H3	3 часа
TIMEFRAME_H4	4 часа
TIMEFRAME_H6	6 часов
TIMEFRAME_H8	8 часов
TIMEFRAME_H12	12 часов
TIMEFRAME_D1	1 день
TIMEFRAME_W1	1 неделя
TIMEFRAME_MN1	1 месяц

Пример:

```
from datetime import datetime
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# импортируем модуль pandas для вывода полученных данных в табличной форме
import pandas as pd
pd.set_option('display.max_columns', 500) # сколько столбцов показываем
pd.set_option('display.width', 1500)      # макс. ширина таблицы для показа
# импортируем модуль pytz для работы с таймзоной
```

© 2000-2025, MetaQuotes Ltd.

```
import pytz

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# установим таймзону в UTC
timezone = pytz.timezone("Etc/UTC")
# создадим объект datetime в таймзоне UTC, чтобы не применялось смещение локальной таймзоны
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# получим 10 баров с EURUSD H4 начиная с 01.10.2020 в таймзоне UTC
rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_H4, utc_from, 10)

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
# выведем каждый элемент полученных данных на новой строке
print("Выведем полученные данные как есть")
for rate in rates:
    print(rate)

# создадим из полученных данных DataFrame
rates_frame = pd.DataFrame(rates)
# сконвертируем время в виде секунд в формат datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# выведем данные
print("\nВыведем датафрейм с данными")
print(rates_frame)

Результат:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Выведем полученные данные как есть
(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)

Выведем датафрейм с данными
                 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
```

© 2000-2025, MetaQuotes Ltd.

```
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
```

Смотри также

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_rates_from_pos

Получает бары из терминала MetaTrader 5, начиная с указанного индекса.

```
copy_rates_from_pos(
   symbol,       // имя символа
   timeframe,    // таймфрейм
   start_pos,    // номер начального бара
   count         // количество баров
   )
```

Параметры

symbol

[in] Имя финансового инструмента, например, "EURUSD". Обязательный неименованный параметр.

timeframe

[in] Таймфрейм, для которого запрашиваются бары. Задается значением из перечисления TIMEFRAME. Обязательный неименованный параметр.

start_pos

[in] Начальный номер бара, с которого запрашиваются данные. Нумерация баров идет от настоящего к прошлому, то есть нулевой бар означает текущий. Обязательный неименованный параметр.

count

[in] Количество баров, которое необходимо получить. Обязательный неименованный параметр.

Возвращаемое значение

Возвращает бары в виде массива numpy с именованными столбцами time, open, high, low, close, tick_volume, spread и real_volume.В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Для дополнительной информации смотрите функцию CopyRates().

Терминал MetaTrader 5 отдает бары только в пределах истории, доступной пользователю на графиках. Количество баров, которые доступны пользователю, задается в настройках параметром "Макс. баров в окне".

Пример:

```
from datetime import datetime
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# импортируем модуль pandas для вывода полученных данных в табличной форме
```

© 2000-2025, MetaQuotes Ltd.

```
import pandas as pd
pd.set_option('display.max_columns', 500) # сколько столбцов показываем
pd.set_option('display.width', 1500)      # макс. ширина таблицы для показа

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# запросим 10 баров на GBPUSD D1 с текущего дня
rates = mt5.copy_rates_from_pos("GBPUSD", mt5.TIMEFRAME_D1, 0, 10)

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()

# выведем каждый элемент полученных данных на новой строке
print("Выведем полученные данные как есть")
for rate in rates:
    print(rate)

# создадим из полученных данных DataFrame
rates_frame = pd.DataFrame(rates)
# сконвертируем время в виде секунд в формат datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# выведем данные
print("\nВыведем датафрейм с данными")
print(rates_frame)

Результат:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Выведем полученные данные как есть
(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)

Выведем датафрейм с данными
        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
```

© 2000-2025, MetaQuotes Ltd.

```
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
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
```

Смотри также

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_rates_range

Получает бары в указанном диапазоне дат из терминала MetaTrader 5.

```
copy_rates_range(
   symbol,        // имя символа
   timeframe,     // таймфрейм
   date_from,     // дата, с которой запрашиваются бары
   date_to        // дата, по которую запрашиваются бары
   )
```

Параметры

symbol

[in] Имя финансового инструмента, например, "EURUSD". Обязательный неименованный параметр.

timeframe

[in] Таймфрейм, для которого запрашиваются бары. Задается значением из перечисления TIMEFRAME. Обязательный неименованный параметр.

date_from

[in] Дата, начиная с которой запрашиваются бары. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Отдаются бары со временем открытия >= date_from. Обязательный неименованный параметр.

date_to

[in] Дата, по которую запрашиваются бары. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Отдаются бары со временем открытия <= date_to. Обязательный неименованный параметр.

Возвращаемое значение

Возвращает бары в виде массива numpy с именованными столбцами time, open, high, low, close, tick_volume, spread и real_volume.В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Для дополнительной информации смотрите функцию CopyRates().

Терминал MetaTrader 5 отдает бары только в пределах истории, доступной пользователю на графиках. Количество баров, которые доступны пользователю, задается в настройках параметром "Макс. баров в окне".

Python при создании объекта datetime использует таймзону локального времени, в то время как терминал MetaTrader 5 хранит время тиков и открытия баров в UTC таймзоне (без смещения). Поэтому, для выполнения функций, использующих время, необходимо создавать datetime в UTC-времени. Данные, полученные из терминала MetaTrader 5, имеют UTC-время.

Пример:

```
from datetime import datetime
import MetaTrader5 as mt5
```

© 2000-2025, MetaQuotes Ltd.

```
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# импортируем модуль pandas для вывода полученных данных в табличной форме
import pandas as pd
pd.set_option('display.max_columns', 500) # сколько столбцов показываем
pd.set_option('display.width', 1500)      # макс. ширина таблицы для показа
# импортируем модуль pytz для работы с таймзоной
import pytz

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# установим таймзону в UTC
timezone = pytz.timezone("Etc/UTC")
# создадим объекты datetime в таймзоне UTC, чтобы не применялось смещение локальной та
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# получим бары с USDJPY M5 в интервале 2020.01.10 00:00 - 2020.01.11 13:00 в таймзоне
rates = mt5.copy_rates_range("USDJPY", mt5.TIMEFRAME_M5, utc_from, utc_to)

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
# выведем каждый элемент полученных данных на новой строке
print("Выведем полученные данные как есть")
counter=0
for rate in rates:
    counter+=1
    if counter<=10:
        print(rate)

# создадим из полученных данных DataFrame
rates_frame = pd.DataFrame(rates)
# сконвертируем время в виде секунд в формат datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# выведем данные
print("\nВыведем датафрейм с данными")
print(rates_frame.head(10))
```

Релультат:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Выведем полученные данные как есть
(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)
```

© 2000-2025, MetaQuotes Ltd.

```
(1578616500, 109.51, 109.51, 109.491, 109.496, 44, 8, 0)
(1578616800, 109.496, 109.509, 109.487, 109.5, 85, 5, 0)
(1578617100, 109.5, 109.504, 109.487, 109.489, 82, 7, 0)

Выведем датафрейм с данными
                 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
```

Смотри также

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_ticks_from

Получает тики из терминала MetaTrader 5, начиная с указанной даты.

```
copy_ticks_from(
   symbol,       // имя символа
   date_from,    // дата, с которой запрашиваются тики
   count,        // количество запрашиваемых тиков
   flags         // комбинация флагов, определяющая тип запрашиваемых тиков
   )
```

Параметры

symbol

[in] Имя финансового инструмента, например, "EURUSD". Обязательный неименованный параметр.

date_from

[in] Дата, начиная с которой запрашиваются тики. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр.

count

[in] Количество тиков, которое необходимо получить. Обязательный неименованный параметр.

flags

[in] Флаг, определяющий тип запрашиваемых тиков. COPY_TICKS_INFO - тики, вызванные изменениями Bid и/или Ask, COPY_TICKS_TRADE - тики с изменения Last и Volume, COPY_TICKS_ALL - все тики. Значения флагов описаны в перечислении COPY_TICKS. Обязательный неименованный параметр.

Возвращаемое значение

Возвращает тики в виде массива numpy с именованными столбцами time, bid, ask, last и flags. Значение flags может быть комбинацией флагов из перечисления TICK_FLAG. В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Для дополнительной информации смотрите функцию CopyTicks.

Python при создании объекта datetime использует таймзону локального времени, в то время как терминал MetaTrader 5 хранит время тиков и открытия баров в UTC таймзоне (без смещения). Поэтому, для выполнения функций, использующих время, необходимо создавать datetime в UTC-времени. Данные, полученные из терминала MetaTrader 5, имеют UTC-время.

COPY_TICKS определяет типы тиков, которые могут быть запрошены с помощью функций copy_ticks_from() и copy_ticks_range().

Идентификатор	Описание
COPY_TICKS_ALL	все тики

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
COPY_TICKS_INFO	тики, содержащие изменения цен Bid и/или Ask
COPY_TICKS_TRADE	тики, содержащие изменения цены Last и/или объема (Volume)

TICK_FLAG определяет возможные флаги для тиков. Данные флаги используются для описания тиков, полученных функциями copy_ticks_from() и copy_ticks_range().

Идентификатор	Описание
TICK_FLAG_BID	изменилась цена Bid
TICK_FLAG_ASK	измeналась цена Ask
TICK_FLAG_LAST	изменилась цена Last
TICK_FLAG_VOLUME	изменился объем (Volume)
TICK_FLAG_BUY	изменилась цена последней покупки (Buy)
TICK_FLAG_SELL	изменилась цена последней продажи (Sell)

Пример:

```
from datetime import datetime
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# импортируем модуль pandas для вывода полученных данных в табличной форме
import pandas as pd
pd.set_option('display.max_columns', 500) # сколько столбцов показываем
pd.set_option('display.width', 1500)      # макс. ширина таблицы для показа
# импортируем модуль pytz для работы с таймзоной
import pytz

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# установим таймзону в UTC
timezone = pytz.timezone("Etc/UTC")
# создадим объект datetime в таймзоне UTC, чтобы не применялось смещение локальной таймзоны
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# запросим 100000 тиков по EURUSD с 10.01.2019 в таймзоне UTC
ticks = mt5.copy_ticks_from("EURUSD", utc_from, 100000, mt5.COPY_TICKS_ALL)
print("Получено тиков:",len(ticks))

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()

# выведем данные каждого тика на новой строке
print("Выведем полученные тики как есть")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# создадим из полученных данных DataFrame
ticks_frame = pd.DataFrame(ticks)
# сконвертируем время в виде секунд в формат datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# выведем данные
print("\nВыведем датафрейм с тиками")
print(ticks_frame.head(10))
```

```
Результат:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Получено тиков: 100000
Выведем полученные тики как есть
(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.)

Выведем датафрейм с тиками
                 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          0
1 2020-01-10 00:00:02  1.11049  1.11067   0.0       0  1578614402025    134          0
2 2020-01-10 00:00:04  1.11050  1.11066   0.0       0  1578614404057    134          0
3 2020-01-10 00:00:04  1.11049  1.11067   0.0       0  1578614404344    134          0
```

© 2000-2025, MetaQuotes Ltd.

```
4 2020-01-10 00:00:12  1.11052  1.11064   0.0       0  1578614412106    134          0
5 2020-01-10 00:00:18  1.11039  1.11051   0.0       0  1578614418265    134          0
6 2020-01-10 00:00:18  1.11040  1.11050   0.0       0  1578614418905    134          0
7 2020-01-10 00:00:19  1.11039  1.11051   0.0       0  1578614419519    134          0
8 2020-01-10 00:00:56  1.11037  1.11065   0.0       0  1578614456011    134          0
9 2020-01-10 00:00:56  1.11039  1.11051   0.0       0  1578614456015    134          0
```

Смотри также

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_ticks_range

Получает тики за указанный диапазон дат из терминала MetaTrader 5.

```
copy_ticks_range(
   symbol,       // имя символа
   date_from,    // дата, с которой запрашиваются тики
   date_to,      // дата, по которую запрашиваются тики
   flags         // комбинация флагов, определяющая тип запрашиваемых тиков
   )
```

Параметры

symbol

[in] Имя финансового инструмента, например, "EURUSD". Обязательный неименованный параметр.

date_from

[in] Дата, начиная с которой запрашиваются тики. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр.

date_to

[in] Дата, по которую запрашиваются тики. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр.

flags

[in] Флаг, определяющий тип запрашиваемых тиков. COPY_TICKS_INFO - тики, вызванные изменениями Bid и/или Ask, COPY_TICKS_TRADE - тики с изменения Last и Volume, COPY_TICKS_ALL - все тики. Значения флагов описаны в перечислении COPY_TICKS. Обязательный неименованный параметр.

Возвращаемое значение

Возвращает тики в виде массива numpy с именованными столбцами time, bid, ask, last и flags. Значение flags может быть комбинацией флагов из перечисления TICK_FLAG. В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Для дополнительной информации смотрите функцию CopyTicks.

Python при создании объекта datetime использует таймзону локального времени, в то время как терминал MetaTrader 5 хранит время тиков и открытия баров в UTC таймзоне (без смещения). Поэтому, для выполнения функций, использующих время, необходимо создавать datetime в UTC-времени. Данные, полученные из терминала MetaTrader 5, имеют UTC-время, но при попытке вывести их на печать, Python опять применит смещение для локального времени. Поэтому, полученные данные также необходимо корректировать для визуального представления.

Пример:

```
from datetime import datetime
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
```

© 2000-2025, MetaQuotes Ltd.

```
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# импортируем модуль pandas для вывода полученных данных в табличной форме
import pandas as pd
pd.set_option('display.max_columns', 500) # сколько столбцов показываем
pd.set_option('display.width', 1500)      # макс. ширина таблицы для показа
# импортируем модуль pytz для работы с таймзоной
import pytz

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# установим таймзону в UTC
timezone = pytz.timezone("Etc/UTC")
# создадим объекты datetime в таймзоне UTC, чтобы не применялось смещение локальной та
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, tzinfo=timezone)
# запросим тики по AUDUSD в интервале 11.01.2020 - 11.01.2020
ticks = mt5.copy_ticks_range("AUDUSD", utc_from, utc_to, mt5.COPY_TICKS_ALL)
print("Получено тиков:",len(ticks))

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
# выведем данные каждого тика на новой строке
print("Выведем полученные тики как есть")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# создадим из полученных данных DataFrame
ticks_frame = pd.DataFrame(ticks)
# сконвертируем время в виде секунд в формат datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# выведем данные
print("\nВыведем датафрейм с тиками")
print(ticks_frame.head(10))
```

```
Результат:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Получено тиков: 37008
Выведем полученные тики как есть
(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.)
```

© 2000-2025, MetaQuotes Ltd.

```
(1578614430, 0.68576, 0.68594, 0., 0, 1578614430188, 4, 0.)
(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.)

Выведем датафрейм с тиками
                 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          0
1 2020-01-10 00:00:01  0.68578  0.68594   0.0       0  1578614401128    130          0
2 2020-01-10 00:00:01  0.68575  0.68594   0.0       0  1578614401128    130          0
3 2020-01-10 00:00:11  0.68576  0.68594   0.0       0  1578614411388    130          0
4 2020-01-10 00:00:11  0.68575  0.68594   0.0       0  1578614411560    130          0
5 2020-01-10 00:00:14  0.68576  0.68595   0.0       0  1578614414973    134          0
6 2020-01-10 00:00:30  0.68576  0.68594   0.0       0  1578614430188      4          0
7 2020-01-10 00:00:50  0.68576  0.68595   0.0       0  1578614450408      4          0
8 2020-01-10 00:00:50  0.68576  0.68594   0.0       0  1578614450519      4          0
9 2020-01-10 00:00:56  0.68575  0.68594   0.0       0  1578614456363    130          0
```

Смотри также

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

orders_total

Получает количество действующих ордеров.

```
orders_total()
```

Возвращаемое значение

Целочисленное значение.

Примечание

Функция является аналогом OrdersTotal.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# проверим наличие действующих ордеров
orders=mt5.orders_total()
if orders>0:
    print("Total orders=",orders)
else:
    print("Orders not found")

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

Смотри также

orders_get, positions_total

© 2000-2025, MetaQuotes Ltd.

orders_get

Получает действующие ордеры с возможность фильтрации по символу или тикету. Существует 3 варианта вызова.

Вызов без параметров. Возвращает действующие ордеры по всем символам.

```
orders_get()
```

Вызов с указанием символа, по которому нужно получить действующие ордеры.

```
orders_get(
   symbol="SYMBOL"       // имя символа
)
```

Вызов с указанием группы символов, по которым нужно получить действующие ордеры.

```
orders_get(
   group="GROUP"         // фильтр для отбора ордеров по символам
)
```

Вызов с указанием тикета ордера.

```
orders_get(
   ticket=TICKET         // тикет
)
```

symbol="SYMBOL"

[in] Имя символа. Необязательный именованный параметр. Если символ указан, то параметр *ticket* игнорируется.

group="GROUP"

[in] Фильтр для отбора группы только необходимых символов. Необязательный именованный параметр. Если группа задана, то функция вернет только те действующие ордеры, которые удовлетворяют заданному критерию для имени символа.

ticket=TICKET

[in] Тикет ордера (ORDER_TICKET). Необязательный именованный параметр.

Возвращаемое значение

Возвращает информацию в виде структуры именованных кортежей (namedtuple). В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Функция позволяет получить за один вызов все действующие ордера, аналогом является связка OrdersTotal и OrderSelect.

Параметр *group* позволяет отфильтровать ордеры по символам, допустимо использовать '*' в начале и конце строки.

Параметр *group* может содержать несколько условий, разделенных запятыми. Условие можно задавать как маску с использованием '*'. Для исключения можно использовать символ логического отрицания '!'. При этом все условия применяются последовательно, то есть сначала необходимо указать условия включения в группу, а затем условие исключения. Например,

© 2000-2025, MetaQuotes Ltd.

group="*, !EUR" означает, что сначала нужно выбрать ордеры по всем символам и затем исключить те из них, что содержат в имени символа "EUR".

Пример:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # сколько столбцов показываем
pd.set_option('display.width', 1500)      # макс. ширина таблицы для показа
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# выведем информацию о действующих ордерах на символе 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))
    # выведем все действующие ордеры
    for order in orders:
        print(order)
print()

# получим список ордеров на символах, чьи имена содержат "*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)))
    # выведем эти ордеры в виде таблицы с помощью 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)

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()

Результат:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Total orders on GBPUSD: 2
```

© 2000-2025, 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
```

Смотри также

orders_total, positions_get

© 2000-2025, MetaQuotes Ltd.

order_calc_margin

Возвращает размер маржи в валюте счета для проведения указанной торговой операции.

```
order_calc_margin(
   action,      // тип ордера (ORDER_TYPE_BUY или ORDER_TYPE_SELL)
   symbol,      // имя символа
   volume,      // объем
   price        // цена открытия
   )
```

Параметры

action

[in] Тип ордера, может принимать значения из перечисления ORDER_TYPE. Обязательный неименованный параметр.

symbol

[in] Имя финансового инструмента. Обязательный неименованный параметр.

volume

[in] Объем торговой операции. Обязательный неименованный параметр.

price

[in] Цена открытия. Обязательный неименованный параметр.

Возвращаемое значение

Вещественное значение в случае успешного выполнения, иначе None. Информацию об ошибке можно получить с помощью last_error().

Примечание

Функция позволяет оценить размер маржи, необходимой для указанного типа ордера на текущем счете и при текущем рыночном окружении без учета текущих отложенных ордеров и открытых позиций. Является аналогом OrderCalcMargin.

ORDER_TYPE

Идентификатор	Описание
ORDER_TYPE_BUY	Рыночный ордер на покупку
ORDER_TYPE_SELL	Рыночный ордер на продажу
ORDER_TYPE_BUY_LIMIT	Отложенный ордер Buy Limit
ORDER_TYPE_SELL_LIMIT	Отложенный ордер Sell Limit
ORDER_TYPE_BUY_STOP	Отложенный ордер Buy Stop
ORDER_TYPE_SELL_STOP	Отложенный ордер Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	По достижении цены ордера выставляется отложенный ордер Buy Limit по цене StopLimit

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
ORDER_TYPE_SELL_STOP_LIMIT	По достижении цены ордера выставляется отложенный ордер Sell Limit по цене StopLimit
ORDER_TYPE_CLOSE_BY	Ордер на закрытие позиции встречной позицией

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# получим валюту счета
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# составим список символов
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_currer
    else:
        print("order_calc_margin failed: , error code =", mt5.last_error())

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()


Результат:
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')
```

© 2000-2025, 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
```

Смотри также

order_calc_profit, order_check

© 2000-2025, MetaQuotes Ltd.

order_calc_profit

Возвращает размер прибыли в валюте счета для указанной торговой операции.

```
order_calc_profit(
   action,          // тип ордера (ORDER_TYPE_BUY или ORDER_TYPE_SELL)
   symbol,          // имя символа
   volume,          // объем
   price_open,      // цена открытия
   price_close      // цена закрытия
   );
```

Параметры

action

[in] Тип ордера, может принимать одно из двух значений перечисления ORDER_TYPE: ORDER_TYPE_BUY или ORDER_TYPE_SELL. Обязательный неименованный параметр.

symbol

[in] Имя финансового инструмента. Обязательный неименованный параметр.

volume

[in] Объем торговой операции. Обязательный неименованный параметр.

price_open

[in] Цена открытия. Обязательный неименованный параметр.

price_close

[in] Цена закрытия. Обязательный неименованный параметр.

Возвращаемое значение

Вещественное значение в случае успешного выполнения, иначе None. Информацию об ошибке можно получить с помощью last_error().

Примечание

Функция позволяет оценить результат торговой операции на текущем счете и при текущем рыночном окружении. Является аналогом OrderCalcProfit.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# получим валюту счета
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# составим список символов
symbols = ("EURUSD","GBPUSD","USDJPY")
print("Symbols to check margin:", symbols)
# оценим значения прибыли для покупок и продаж
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_erro
    print()

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

```
Результат:
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
```

© 2000-2025, 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
```

Смотри также

order_calc_margin, order_check

© 2000-2025, MetaQuotes Ltd.

order_check

Проверяет достаточность средств для совершения требуемой торговой операции. Результаты проверки возвращаются в виде структуры MqlTradeCheckResult.

```
order_check(
   request      // структура запроса
   );
```

Параметры

request

[in] Структура типа MqlTradeRequest, которая описывает требуемое торговое действие. Обязательный неименованный параметр. Пример заполнения запроса и состав перечислений смотрите ниже.

Возвращаемое значение

Результат проверки в виде структуры MqlTradeCheckResult. Поле *request* в ответе содержит структуру торгового запроса, переданного в order_check().

Примечание

Успешная проверка запроса **не является свидетельством того, что запрашиваемая торговая операция непременно выполнится успешно**. Функция order_check является аналогом OrderCheck.

TRADE_REQUEST_ACTIONS

Идентификатор	Описание
TRADE_ACTION_DEAL	Установить торговый ордер на немедленное совершение сделки с указанными параметрами (поставить рыночный ордер)
TRADE_ACTION_PENDING	Установить торговый ордер на совершение сделки при указанных условиях (отложенный ордер)
TRADE_ACTION_SLTP	Изменить значения Stop Loss и Take Profit у открытой позиции
TRADE_ACTION_MODIFY	Изменить параметры ранее установленного торгового ордера
TRADE_ACTION_REMOVE	Удалить ранее выставленный отложенный торговый ордер
TRADE_ACTION_CLOSE_BY	Закрыть позицию встречной

ORDER_TYPE_FILLING

Идентификатор	Описание
ORDER_FILLING_FOK	Данная политика исполнения означает, что ордер может быть исполнен исключительно в указанном объеме. Если на рынке в данный момент не присутствует достаточного

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
	объема финансового инструмента, то ордер не будет исполнен. Необходимый объем может быть составлен из нескольких предложений, доступных в данный момент на рынке.
ORDER_FILLING_IOC	Означает согласие совершить сделку по максимально доступному на рынке объему в пределах указанного в ордере. В случае невозможности полного исполнения ордер будет исполнен на доступный объем, а неисполненный объем ордера будет отменен.
ORDER_FILLING_RETURN	Данный режим используется для рыночных (ORDER_TYPE_BUY и ORDER_TYPE_SELL), лимитных и стоп-лимитных ордеров (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT и ORDER_TYPE_SELL_STOP_LIMIT) и только в режимах "Исполнение по рынку" и "Биржевое исполнение". В случае частичного исполнения рыночный или лимитный ордер с остаточным объемом не снимается, а продолжает действовать. Для ордеров ORDER_TYPE_BUY_STOP_LIMIT и ORDER_TYPE_SELL_STOP_LIMIT при активации будет создан соответствующий лимитный ордер ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT с типом исполнения ORDER_FILLING_RETURN.

ORDER_TYPE_TIME

Идентификатор	Описание
ORDER_TIME_GTC	Ордер будет находиться в очереди до тех пор, пока не будет снят
ORDER_TIME_DAY	Ордер будет действовать только в течение текущего торгового дня
ORDER_TIME_SPECIFIED	Ордер будет действовать до даты истечения
ORDER_TIME_SPECIFIED_DAY	Ордер будет действовать до 23:59:59 указанного дня. Если это время не попадает на торговую сессию, истечение наступит в ближайшее торговое время.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# получим валюту счета
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# подготовим структуру запроса
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()

# если символ недоступен в MarketWatch, добавим его
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()

# подготовим запрос
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,
}

# выполним проверку и выведем результат как есть
result = mt5.order_check(request)
print(result);
# запросим результат в виде словаря и выведем поэлементно
result_dict=result._asdict()
for field in result_dict.keys():
    print("   {}={}".format(field,result_dict[field]))
    # если это структура торгового запроса, то выведем её тоже поэлементно
    if field=="request":
        traderequest_dict=result_dict[field]._asdict()
        for tradereq_filed in traderequest_dict:
            print("       traderequest: {}={}".format(tradereq_filed,traderequest_dict
```

© 2000-2025, MetaQuotes Ltd.

```
# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()


Результат:
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
```

Смотри также

order_send, OrderCheck, Типы торговых операций, Структура торгового запроса, Структура результатов проверки торгового запроса, Структура результата торгового запроса

© 2000-2025, MetaQuotes Ltd.

order_send

Отправляет из терминала на торговый сервер запрос на совершение торговой операции. Функция является аналогом OrderSend.

```
order_send(
   request      // структура запроса
   );
```

Параметры

request

[in] Структура типа MqlTradeRequest, которая описывает требуемое торговое действие. Обязательный неименованный параметр. Пример заполнения запроса и состав перечислений смотрите ниже.

Возвращаемое значение

Результат выполнения в виде структуры MqlTradeResult. Поле *request* в ответе содержит структуру торгового запроса, переданного в order_send().В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Структура торгового запроса MqlTradeRequest

Поле	Описание
action	Тип торговой операции. Значение может быть одним из значений перечисления TRADE_REQUEST_ACTIONS
magic	Идентификатор эксперта. Позволяет организовать аналитическую обработку торговых ордеров. Каждый эксперт может выставлять свой собственный уникальный идентификатор при отправке торгового запроса
order	Тикет ордера. Требуется для модификации отложенных ордеров
symbol	Имя торгового инструмента, по которому выставляется ордер. Не требуется при операциях модификации ордеров и закрытии позиций
volume	Запрашиваемый объем сделки в лотах. Реальное значение объема при открытии сделки будет зависеть от типа ордера по исполнению.
price	Цена, при достижении которой ордер должен быть исполнен. Для рыночных ордеров по инструментам с типом исполнения "Market Execution" (SYMBOL_TRADE_EXECUTION_MARKET), имеющих тип TRADE_ACTION_DEAL, указание цены не требуется
stoplimit	Цена, по которой будет выставлен отложенный Limit-ордер при достижении ценой значения price (это условие является обязательным). До этого момента отложенный ордер в торговую систему не выводится
sl	Цена, по которой сработает Stop Loss ордер при движении цены в неблагоприятном направлении
tp	Цена, по которой сработает Take Profit ордер при движении цены в благоприятном направлении

© 2000-2025, MetaQuotes Ltd.

Поле	Описание
deviation	Максимально приемлемое отклонение от запрашиваемой цены, задаваемое в пунктах
type	Тип ордера. Значение может быть одним из значений перечисления ORDER_TYPE
type_filling	Тип ордера по исполнению. Значение может быть одним из значений ORDER_TYPE_FILLING
type_time	Тип ордера по истечению. Значение может быть одним из значений ORDER_TYPE_TIME
expiration	Срок истечения отложенного ордера (для ордеров типа TIME_SPECIFIED)
comment	Комментарий к ордеру
position	Тикет позиции. Следует заполнять при изменении и закрытии позиции для ее однозначной идентификации. Как правило, соответствует тикету ордера, в результате которого позиция была открыта.
position_by	Тикет встречной позиции. Используется при закрытии позиции встречной — открытой по тому же инструменту, но в противоположном направлении.

Примечание

Торговый запрос проходит несколько стадий проверок на торговом сервере. В первую очередь проверяется корректность заполнения всех необходимых полей запроса *request*, и при отсутствии ошибок сервер принимает ордер для дальнейшей обработки. Полную информацию по исполнению торговых операций смотрите в описании функции OrderSend.

Пример:

```
import time
import MetaTrader5 as mt5

# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ", mt5.__author__)
print("MetaTrader5 package version: ", mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# подготовим структуру запроса для покупки
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()
```

© 2000-2025, MetaQuotes Ltd.

```
# если символ недоступен в MarketWatch, добавим его
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,
}

# отправим торговый запрос
result = mt5.order_send(request)
# проверим результат выполнения
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))
    # запросим результат в виде словаря и выведем поэлементно
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # если это структура торгового запроса, то выведем её тоже поэлементно
        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))
```

© 2000-2025, MetaQuotes Ltd.

```
print("   sleep 2 seconds before closing position #{}".format(result.order))
time.sleep(2)
# создадим запрос на закрытие
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,
}
# отправим торговый запрос
result=mt5.order_send(request)
# проверим результат выполнения
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))
    # запросим результат в виде словаря и выведем поэлементно
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # если это структура торгового запроса, то выведем её тоже поэлементно
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

```
Результат
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
```

© 2000-2025, MetaQuotes Ltd.

```
4. position #557416535 closed, OrderSendResult(retcode=10009, deal=535084631, order=55
   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
```

Смотри также

order_check, OrderSend,Типы торговых операций, Структура торгового запроса, Структура результатов проверки торгового запроса, Структура результата торгового запроса

© 2000-2025, MetaQuotes Ltd.

positions_total

Получает количество открытых позиций.

```
positions_total()
```

Возвращаемое значение

Целочисленное значение.

Примечание

Функция является аналогом PositionsTotal.

Пример:

```
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# проверим наличие открытых позиций
positions_total=mt5.positions_total()
if positions_total>0:
    print("Total positions=",positions_total)
else:
    print("Positions not found")

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

Смотри также

positions_get, orders_total

© 2000-2025, MetaQuotes Ltd.

positions_get

Получает открытые позиции с возможностью фильтрации по символу или тикету. Существует 3 варианта вызова.

Вызов без параметров. Возвращает открытые позиции по всем символам.

```
positions_get()
```

Вызов с указанием символа, по которому нужно получить открытые позиции.

```
positions_get(
   symbol="SYMBOL"       // имя символа
)
```

Вызов с указанием группы символов, по которым нужно получить открытые позиции.

```
positions_get(
   group="GROUP"         // фильтр для отбора позиций по символам
)
```

Вызов с указанием тикета позиции.

```
positions_get(
   ticket=TICKET         // тикет
)
```

Параметры

symbol="SYMBOL"

[in] Имя символа. Необязательный именованный параметр. Если символ указан, то параметр *ticket* игнорируется.

group="GROUP"

[in] Фильтр для отбора группы только необходимых символов. Необязательный именованный параметр. Если группа задана, то функция вернет только те позиции, которые удовлетворяют заданному критерию для имени символа.

ticket=TICKET

[in] Тикет позиции (POSITION_TICKET). Необязательный именованный параметр.

Возвращаемое значение

Возвращает информацию в виде структуры именованных кортежей (namedtuple). В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Функция позволяет получить за один вызов все открытые позиции, аналогом является связка PositionsTotal и PositionSelect.

Параметр *group* может содержать несколько условий, разделенных запятыми. Условие можно задавать как маску с использованием '*'. Для исключения можно использовать символ логического отрицания '!'. При этом все условия применяются последовательно, то есть сначала необходимо указать условия включения в группу, а затем условие исключения. Например,

© 2000-2025, MetaQuotes Ltd.

group="*, !EUR" означает, что сначала нужно выбрать позиции по всем символам и затем исключить те из них, что содержат в имени символа "EUR".

Пример:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # сколько столбцов показываем
pd.set_option('display.width', 1500)      # макс. ширина таблицы для показа
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# получим открытые позиции на 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))
    # выведем все открытые позиции
    for position in positions:
        print(position)

# получим список позиций на символах, чьи имена содержат "*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)))
    # выведем эти позиции в виде таблицы с помощью 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)

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()


Результат:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

positions_get(group="*USD*")=5
```

© 2000-2025, 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

Смотри также

positions_total, orders_get

© 2000-2025, MetaQuotes Ltd.

history_orders_total

Получает количество ордеров в торговой истории в указанном интервале.

```
history_orders_total(
   date_from,     // дата, с которой запрашиваются ордеры
   date_to        // дата, по которую запрашиваются ордеры
   )
```

Параметры

date_from

[in] Дата, начиная с которой запрашиваются ордеры. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр.

date_to

[in] Дата, по которую запрашиваются ордеры. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр.

Возвращаемое значение

Целочисленное значение.

Примечание

Функция является аналогом HistoryOrdersTotal.

Пример:

```
from datetime import datetime
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# получим количество ордеров в истории
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")

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

history_orders_get, history_deals_total

© 2000-2025, MetaQuotes Ltd.

history_orders_get

Получает ордеры из торговой истории с возможностью фильтрации по тикету или позиции. Существует 3 варианта вызова.

Вызов с указанием интервала времени. Возвращает все ордеры, попадающие в заданный интервал.

```
history_orders_get(
   date_from,           // дата, с которой запрашиваются ордеры
   date_to,             // дата, по которую запрашиваются ордеры
   group="GROUP"        // фильтр для отбора ордеров по символам
   )
```

Вызов с указанием тикета ордера. Возвращает ордер с указанным тикетом.

```
history_orders_get(
   ticket=TICKET        // тикет ордера
)
```

Вызов с указанием тикета позиции. Возвращает все ордеры, имеющие в свойстве ORDER_POSITION_ID указанный тикет позиции.

```
history_orders_get(
   position=POSITION    // тикет позиции
)
```

Параметры

date_from

[in] Дата, начиная с которой запрашиваются ордеры. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр, указывается первым.

date_to

[in] Дата, по которую запрашиваются ордеры. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр, указывается вторым.

group="GROUP"

[in] Фильтр для отбора группы только необходимых символов. Необязательный именованный параметр. Если группа задана, то функция вернет только те ордеры из истории, которые удовлетворяют заданному критерию для имени символа.

ticket=TICKET

[in] Тикет ордера, который необходимо получить. Необязательный параметр. Если тикет ордера не указан, то фильтр не применяется.

position=POSITION

[in] Тикет позиции (хранится в ORDER_POSITION_ID), для которой необходимо получить все ордеры. Необязательный параметр. Если тикет позиции не указан, то фильтр не применяется.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает информацию в виде структуры именованных кортежей (namedtuple). В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Функция позволяет получить за один вызов все исторические ордеры за указанный период, аналогом является связка HistoryOrdersTotal и HistoryOrderSelect.

Параметр *group* может содержать несколько условий, разделенных запятыми. Условие можно задавать как маску с использованием '*'. Для исключения можно использовать символ логического отрицания '!'. При этом все условия применяются последовательно, то есть сначала необходимо указать условия включения в группу, а затем условие исключения. Например, *group*="*, !EUR" означает, что сначала нужно выбрать сделки по всем символам и затем исключить те из них, что содержат в имени символа "EUR".

Пример:

```
from datetime import datetime
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # сколько столбцов показываем
pd.set_option('display.width', 1500)      # макс. ширина таблицы для показа
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# получим количество ордеров в истории
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()

# выведем все исторические ордеры по тикету позиции
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
    # выведем все исторические ордеры, имеющие указанный тикет позиции
    for position_order in position_history_orders:
```

© 2000-2025, MetaQuotes Ltd.

```
        print(position_order)
    print()
    # выведем эти ордеры в виде таблицы с помощью 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_c
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    df['time_done'] = pd.to_datetime(df['time_done'], unit='s')
    print(df)

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()

Результат:
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
```

Смотри также

history_deals_total, history_deals_get

© 2000-2025, MetaQuotes Ltd.

history_deals_total

Получает количество сделок в торговой истории в указанном интервале.

```
history_deals_total(
   date_from,      // дата, с которой запрашиваются сделки
   date_to         // дата, по которую запрашиваются сделки
   )
```

Параметры

date_from

[in] Дата, начиная с которой запрашиваются сделки. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр.

date_to

[in] Дата, по которую запрашиваются сделки. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр.

Возвращаемое значение

Целочисленное значение.

Примечание

Функция является аналогом HistoryDealsTotal.

Пример:

```
from datetime import datetime
import MetaTrader5 as mt5
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# получим количество сделок в истории
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")

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

Смотри также

history_deals_get, history_orders_total

© 2000-2025, MetaQuotes Ltd.

history_deals_get

Получает сделки из торговой истории в указанном интервале с возможностью фильтрации по тикету или позиции.

Вызов с указанием интервала времени. Возвращает все сделки, попадающие в заданный интервал.

```
history_deals_get(
   date_from,           // дата, с которой запрашиваются сделки
   date_to,             // дата, по которую запрашиваются сделки
   group="GROUP"        // фильтр для отбора сделок по символам
   )
```

Вызов с указанием тикета ордера. Возвращает все сделки, имеющие указанный тикет ордера в свойстве DEAL_ORDER.

```
history_deals_get(
   ticket=TICKET        // тикет ордера
)
```

Вызов с указанием тикета позиции. Возвращает все сделки, имеющие указанный тикет позиции в свойстве DEAL_POSITION_ID.

```
history_deals_get(
   position=POSITION    // тикет позиции
)
```

Параметры

date_from

[in] Дата, начиная с которой запрашиваются ордеры. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр, указывается первым.

date_to

[in] Дата, по которую запрашиваются ордеры. Задается объектом datetime или в виде количества секунд, прошедших с 1970.01.01. Обязательный неименованный параметр, указывается вторым.

group="GROUP"

[in] Фильтр для отбора группы только необходимых символов. Необязательный именованный параметр. Если группа задана, то функция вернет только те сделки, которые удовлетворяют заданному критерию для имени символа.

ticket=TICKET

[in] Тикет ордера (хранится в DEAL_ORDER), для которого необходимо получить все сделки. Необязательный параметр. Если тикет ордера не указан, то фильтр не применяется.

position=POSITION

[in] Тикет позиции (хранится в DEAL_POSITION_ID), для которой необходимо получить все сделки. Необязательный параметр. Если тикет позиции не указан, то фильтр не применяется.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Возвращает информацию в виде структуры именованных кортежей (namedtuple). В случае ошибки возвращает None, информацию об ошибке можно получить с помощью last_error().

Примечание

Функция позволяет получить за один вызов все сделки из истории за указанный период, аналогом является связка HistoryDealsTotal и HistoryDealSelect.

Параметр *group* позволяет отфильтровать сделки по символам, допустимо использовать '*' в начале и конце строки.

Параметр *group* может содержать несколько условий, разделенных запятыми. Условие можно задавать как маску с использованием '*'. Для исключения можно использовать символ логического отрицания '!'. При этом все условия применяются последовательно, то есть сначала необходимо указать условия включения в группу, а затем условие исключения. Например, *group*="*, !EUR" означает, что сначала нужно выбрать сделки по всем символам и затем исключить те из них, что содержат в имени символа "EUR".

Пример:

```
import MetaTrader5 as mt5
from datetime import datetime
import pandas as pd
pd.set_option('display.max_columns', 500) # сколько столбцов показываем
pd.set_option('display.width', 1500)      # макс. ширина таблицы для показа
# выведем данные о пакете MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# установим подключение к терминалу MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# получим количество сделок в истории
from_date=datetime(2020,1,1)
to_date=datetime.now()
# получим  в указанном интервале сделки по символам, имена которых содержат "GBP"
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,len

# получим  сделки по символам, имена которых не содержат ни "EUR" ни "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
    # выведем все полученные сделки как есть
```

© 2000-2025, MetaQuotes Ltd.

```
    for deal in deals:
        print("  ",deal)
    print()
    # выведем эти сделки в виде таблицы с помощью pandas.DataFrame
    df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)
print("")

# получим все сделки, которые относятся к позиции #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)))
    # выведем эти сделки в виде таблицы с помощью 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)

# завершим подключение к терминалу MetaTrader 5
mt5.shutdown()
```

Результат:

```
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=15831762422
   TradeDeal(ticket=516943494, order=539349382, time=1583510003, time_msc=15835100038
   TradeDeal(ticket=516943915, order=539349802, time=1583510025, time_msc=15835100250
   TradeDeal(ticket=517139682, order=539557870, time=1583520201, time_msc=15835202012
   TradeDeal(ticket=517139716, order=539557909, time=1583520202, time_msc=15835202029

      ticket      order                time       time_msc  type  entry  magic  positi
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    5302
2  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    5302
3  516943494  539349382 2020-03-06 15:53:23  1583510003895     1      0      0    5393
4  516943915  539349802 2020-03-06 15:53:45  1583510025054     0      0      0    5393
5  517139682  539557870 2020-03-06 18:43:21  1583520201227     0      1      0    5393
6  517139716  539557909 2020-03-06 18:43:22  1583520202971     1      1      0    5393
```

© 2000-2025, 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
```

Смотри также

history_deals_total, history_orders_get

© 2000-2025, MetaQuotes Ltd.

ONNX модели в машинном обучении

ONNX (Open Neural Network Exchange) — открытая библиотека для построения нейронных сетей глубокого обучения. У данного проекта есть несколько крупных преимуществ:

- Поддержка ONNX осуществляется крупными компаниями, такими как Microsoft, Facebook, Amazon и другими партнерами.
- Открытый формат позволяет производить преобразование форматов между различными средствами машинного обучения, ONNXMLTools от Microsoft позволяет конвертировать модели в формат ONNX.
- MQL5 обеспечивает автоматическую конвертацию входных и выходных данных модели, если тип переданного параметра не соответствует модели.
- Создавать ONNX модели можно с помощью множества инструментов машинного обучения, в настоящее время они поддерживаются в Caffe2, Microsoft Cognitive Toolkit, MXNet, PyTorch и OpenCV, также есть интерфейсы для многих других популярных фреймворков и библиотек.
- Язык MQL5 позволяет внедрить ONNX модель в торговую стратегию и использовать все преимущества платформы MetaTrader 5 для эффективной работы на финансовых рынках.
- Перед запуском модели в реальную торговлю вы можете проверить модель на исторических данных в тестере стратегий без использования сторонних инструментов.

Для работы с ONNX в MQL5 предусмотрены следующие функции:

Функция	Действие
OnnxCreate	Создает сессию ONNX с загрузкой модели из *.onnx файла
OnnxCreateFromBuffer	Создает сессию ONNX с загрузкой модели из массива данных
OnnxRelease	Удаление сессии ONNX
OnnxRun	Запуск модели ONNX на исполнение
OnnxGetInputCount	Получение количества входных параметров модели ONNX
OnnxGetOutputCount	Получение количества выходных параметров модели ONNX
OnnxGetInputName	Получение имени входного параметра модели по индексу
OnnxGetOutputName	Получение имени выходного параметра модели по индексу
OnnxGetInputTypeInfo	Получение описания типа входного параметра у модели
OnnxGetOutputTypeInfo	Получение описания типа выходного параметра у модели
OnnxSetInputShape	Устанавливает размерность входных данных модели по индексу
OnnxSetOutputShape	Устанавливает размерность выходных данных модели по индексу

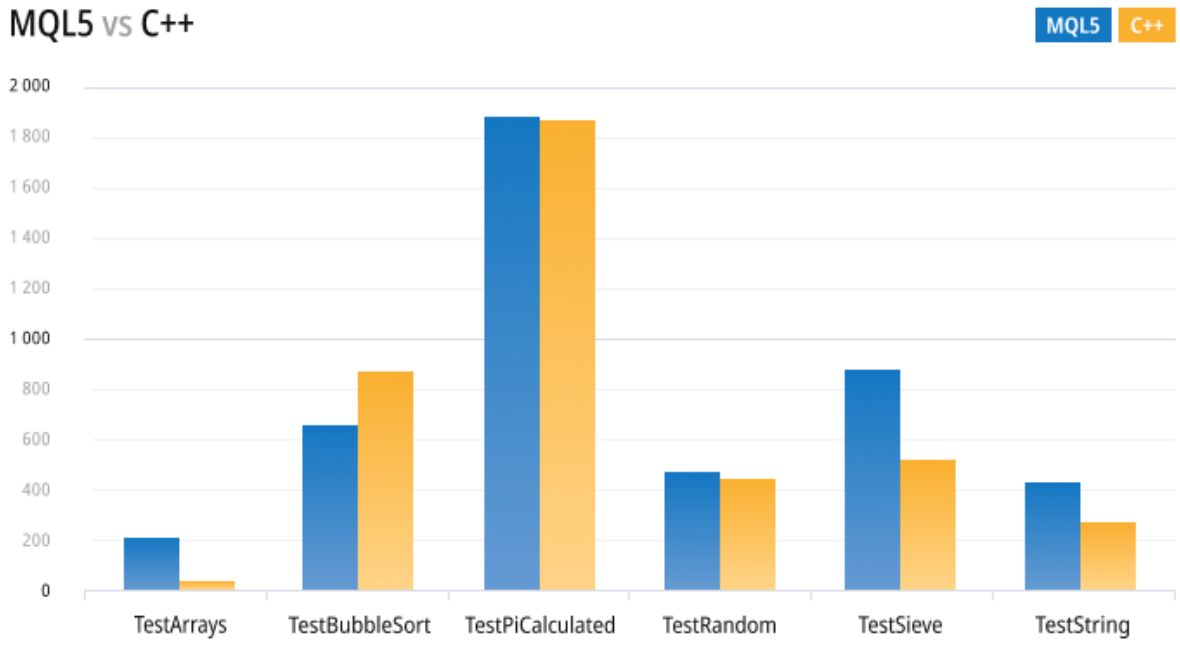
© 2000-2025, MetaQuotes Ltd.

Поддержка ONNX в MQL5

ONNX — это открытый формат, созданный для представления моделей машинного обучения. Данный стандарт определяет общий набор операторов и формат файлов, что позволяет разработчикам использовать модели на различных платформах и средах исполнения, с применением различных инструментов и компиляторов.

Таким образом, открытый формат ONNX позволяет получать и перемещать модели машиннного обучения между разными платформами и инструментами машинного обучения. Именно поэтому поддержка ONNX добавлена в язык MQL5 — разработчики ИИ могут запускать созданные модели в высокопроизводительной среде исполнения на платформе MetaTrader 5.

Язык MQL5 не уступает по скорости выполнения приложениям на C++. Результаты выполнения стандартных тестов на языках MQL5 и C++ подтверждают это. Чем ниже столбик, тем меньше времени затрачено на выполнение, тем лучше результат (время в миллисекундах). Тесты проведены на Windows 10 (build 17763) x64, Xeon E5-2630 v4 @ 2.20GHz, Memory: 65457 Mb.



С новыми асинхронными торговыми операциями и нативной поддержкой ONNX открываются новые возможности, которые раньше были доступны только избранным профессионалам и институциональным трейдерам. Поддержка ONNX в MQL5 позволяет трейдерам обучать модели для финансовых рынках в предпочитаемой среде разработки и затем торговать с низкими сетевыми издержками, на высокой скорости обновления стакана и с использованием асинхронной отсылки приказов.

В данный момент ONNX разрабатывается и поддерживается совместно такими компаниями как Microsoft, Facebook, Amazon и другими партнерами, что дает гарантию дальнейшего развития этого проекта с открытым исходным кодом.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Преобразование форматов

Формат ONNX является открытым, что позволяет сохранять в нем модели из различных средств машинного обучения. Этот формат поддерживается множеством платформ, включая Chainer, Caffee2 и PyTorch.

Одним из наиболее популярных инструментов для преобразовывания моделей в формат ONNX является ONNXMLTools от Microsoft.

Инструкции по установке и использованию ONNXMLTools доступны в репозитории на GitHub. В настоящее время поддерживаются следующие наборы средств:

- Keras (оболочка конвертора keras2onnx)
- Tensorflow (оболочка конвертора tf2onnx)
- scikit-learn (оболочка конвертора skl2onnx)
- Apple Core ML
- Spark ML (экспериментальный режим)
- LightGBM
- libscm;
- XGBoost;
- H2O
- CatBoost

Установка ONNXMLTools проста и описана на странице проекта https://github.com/onnx/onnxmltools#install, там же приведены некоторые примеры конвертации моделей.

© 2000-2025, MetaQuotes Ltd.

Автоконвертация input и output значений при запуске ONNX моделей

В текущей версии ONNX в языке MQL5 в качестве input/output значений поддерживаются только тензоры — массивы данных с одним из следующих типов для элементов:

Тип ONNX	Соответствует типу 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	—

В качестве input/output значений в ONNX модели можно передавать только массивы, вектора или матрицы (далее **Данные**).

Если тип переданного параметра не соответствует типу параметру ONNX модели и при вызове OnnxRun не указан флаг ONNX_NO_CONVERSION, будет использована автоконвертация. Автоконвертация означает, что перед выполнением ONNX модели пользовательские **Данные** будут скопированы в ONNX-тензоры с конвертацией.

При запуске ONNX модели без автоконвертации ONNX модель расчитывается на **Данных** без какого либо дополнительного копирования.

> **ВАЖНО!** Автоконвертация не контролирует переполнение (truncate), поэтому следует внимательно следить за данными и их типами, которые передаются в ONNX модель.

Автоконвертация поддерживает следующие типы ONNX:

© 2000-2025, 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

Неподдерживаемые типы:

- ONNX_DATA_TYPE_BFLOAT16
- ONNX_DATA_TYPE_STRING

Правила автоконвертации по типам тензоров

Если тип MQL5 не входит в список типов, поддерживаемых моделью, то запуск ONNX модели завершится с ошибкой ERR_ONNX_NOT_SUPPORTED (код ошибки 5802).

Примечание: при автоконвертациии тип color обрабатывается как uint, а тип datetime как long.

Автоконвертация input значений

Тип ONNX (тип элемента тензора)	Тип MQL5, поддерживаемый автоконвертацией
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex При конвертации элементы **Данных** проверяются простым сравнением на ноль
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	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	см. ONNX_DATA_TYPE_FLOAT

© 2000-2025, MetaQuotes Ltd.

Тип ONNX (тип элемента тензора)	Тип MQL5, поддерживаемый автоконвертацией
ONNX_DATA_TYPE_UINT16	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

Автоконвертация output значений

Тип ONNX (тип элемента тензора)	Тип MQL5, поддерживаемый автоконвертацией
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex Если элемент тензора ноль, то элементу Данных выставляется значение 0, иначе – значение 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	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT16	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	см. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex

© 2000-2025, MetaQuotes Ltd.

Тип ONNX (тип элемента тензора)	Тип MQL5, поддерживаемый автоконвертацией
ONNX_DATA_TYPE_COMPLEX128	complex

Смотри также

Приведение типов

© 2000-2025, MetaQuotes Ltd.

Создание модели

Есть несколько способов получить готовую модель в формате ONNX. Признанной библиотекой является ONNX Model Zoo, которая содержит несколько предварительно обученных моделей ONNX для разных типов задач. Достоинством данной коллекции является то, что в ноутбуке каждой модели содержатся ссылки на набор обучающих данных, а также ссылки на оригинальный научный документ, описывающий архитектуру модели.

Большинство платформ для машиного обучения использует Python, для установки среды выполнения ONNX для Python, используйте одну из следующих команд:

```
pip install onnxruntime        # CPU build
pip install onnxruntime-gpu    # GPU build
```

Для вызова среды выполнения ONNX в Python используйте следующую команду

```
import onnxruntime
session = onnxruntime.InferenceSession("path to model")
```

Входные и выходные данные для использования модели нужно смотреть в документации к соответствующей модели. Для просмотра модели можно также использовать такие средства визуализации, как Netron или WinML Dashboard. Среда выполнения ONNX также позволяет запрашивать метаданные модели и ее входные и выходные данные:

```
results = session.run(["output1", "output2"], {
                      "input1": indata1, "input2": indata2})
results = session.run([], {"input1": indata1, "input2": indata2})
```

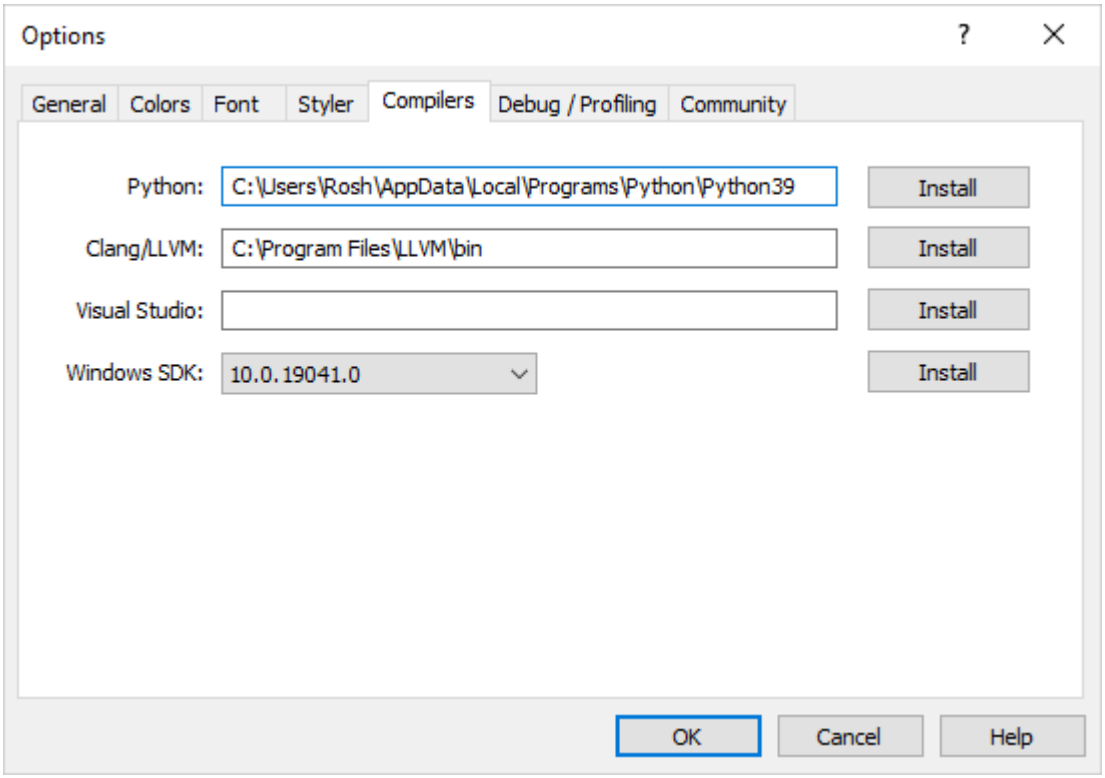
Вы можете создавать ONNX модели прямо в терминале MetaTrader 5 или в среде разработки MetaEditor на Python.

Python в MetaTrader 5

Платформа MetaTrader 5 поддерживает работу со скриптами Python прямо из коробки. Для этого разработчики терминала предоставили и поддерживают модуль MetaTrader5 для Python — https://pypi.org/project/MetaTrader5.

Интегрированная среда разработки MetaEditor позволяет писать не только приложения на MQL5, но и запускать питоновские скрипты прямо из редактора. Для этого необходимо указать путь к исполняемому файлу в настройках MetaEditor:

© 2000-2025, MetaQuotes Ltd.



Если Python отсутствует на вашем компьютере, нажмите "Установить", чтобы загрузить установочный файл.

Вы можете создать в MetaEdtior или загрузить в папку данных терминала скрипт на Python и сразу же запустить его по клавише F7 ("Компилировать"). После этого будет открыт терминал MetaTrader 5 и на текущем графике будет запущен скрипт. Сообщения из консоли Python (stdout, stderr) будут выведены в раздел "Ошибки".

Схема работы с моделями в MetaTrader 5

Язык MQL5 позволяет запускать ONNX модели прямо в терминале MetaTrader 5. Делается это в 3 шага:

1. Натренировать модель в сторонней платформе, например, в Python.
2. Сконвертировать модель в ONNX
3. Включить ONNX модель в советника с помощью ONNX-функций и запустить в терминале MetaTrader 5.

Python интеграция языка MQL5 позволяет запускать питоновский скрипт и сохранять ONNX модель в редакторе MetaEditor или даже запуском на графике в MetaTrader 5. Вы можете тренировать модель с помощью однажды написанного скрипта на Python с нужной вам периодичностью прямо в терминале. Так как библиотека содержит встроенные готовые функции для получения ценовых данных, которые можно подать на вход ONNX модели:

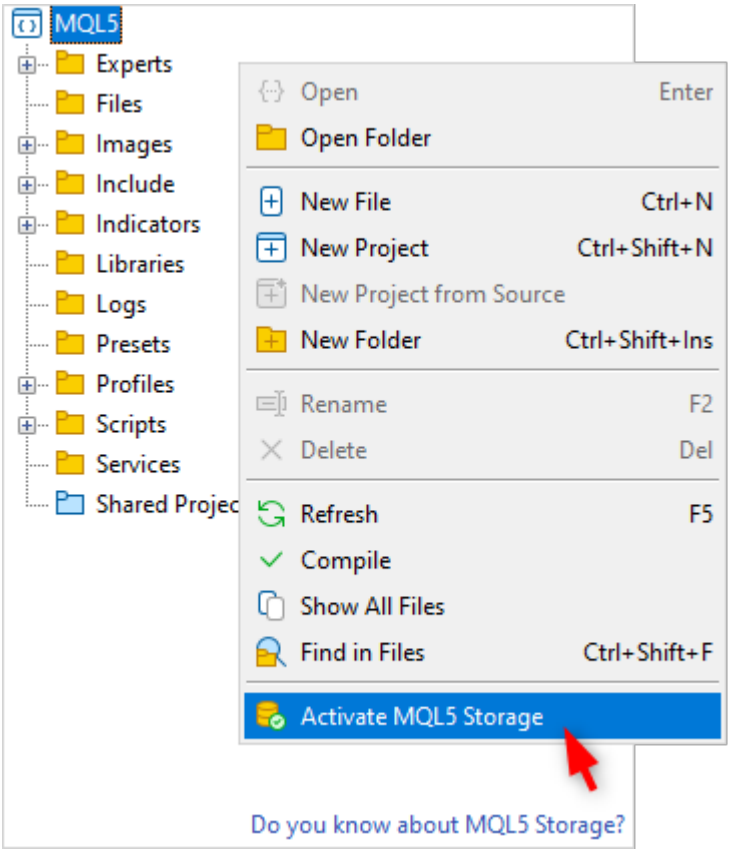
- copy_rates_from - получает бары, начиная с указанной даты;
- copy_rates_from_pos - получает бары, начиная с указанного индекса;
- copy_rates_range - получает бары в указанном диапазоне дат;
- copy_ticks_from - получает тики, начиная с указанной даты;

© 2000-2025, MetaQuotes Ltd.

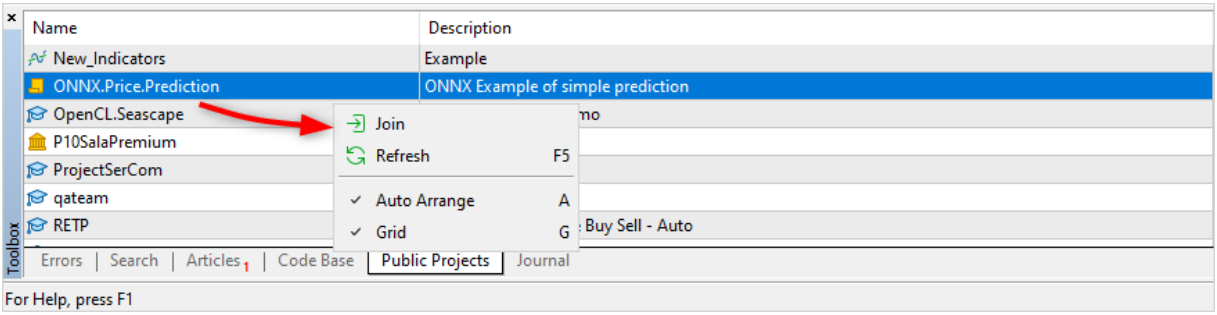
- copy_ticks_range - получает тики за указанный диапазон дат.

Пример готовой модели

Пример готовой ONNX модели вы можете найти в публичных проектах. Для этого необходимо предварительно активировать MQL5 Storage в Навигаторе, указав свой MQL5 логин в настройках MetaEditor с учетом регистра.

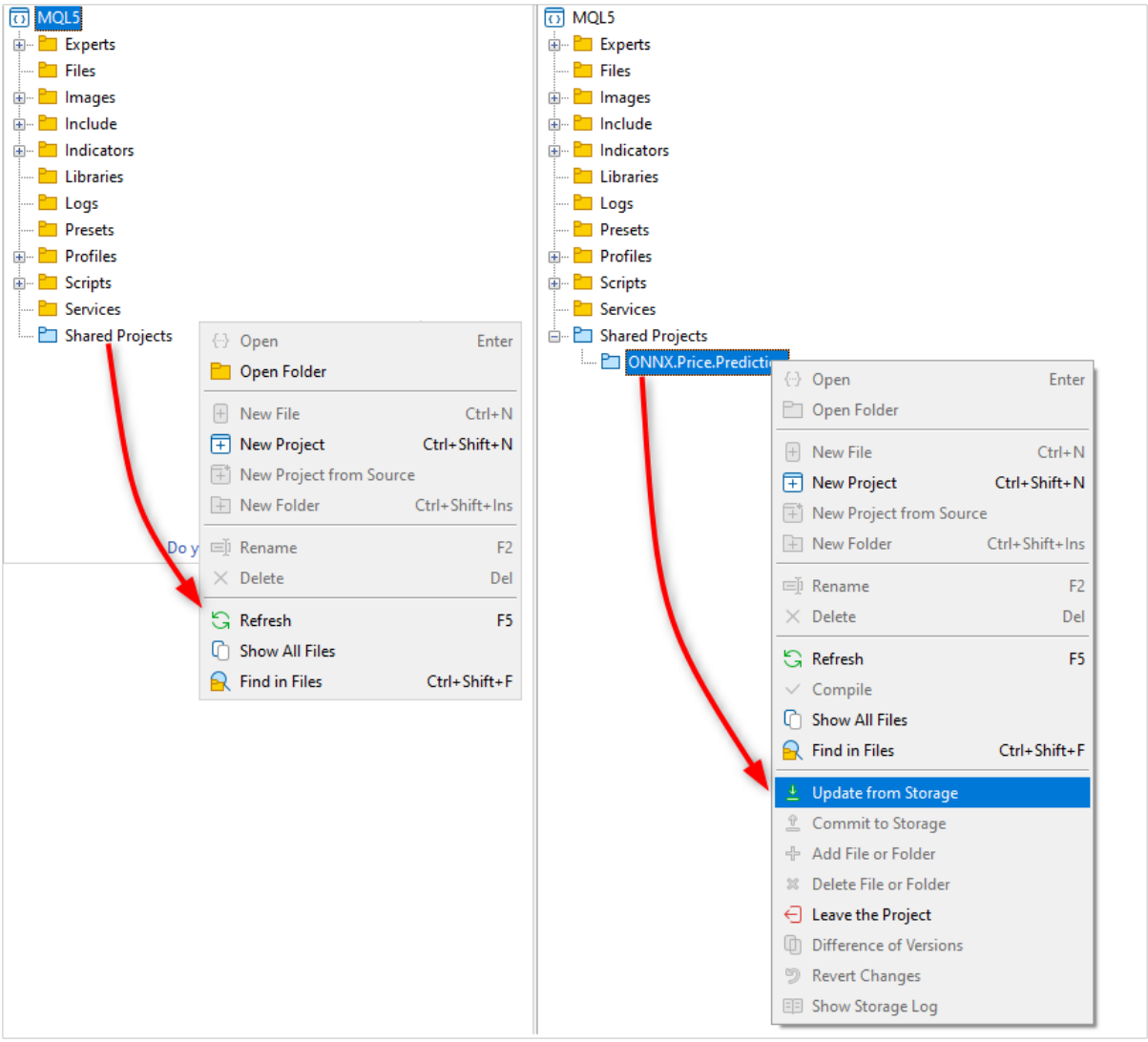


После активации найдите проект ONNX.Price.Prediction и присоединитесь к нему через команду контекстного меню.



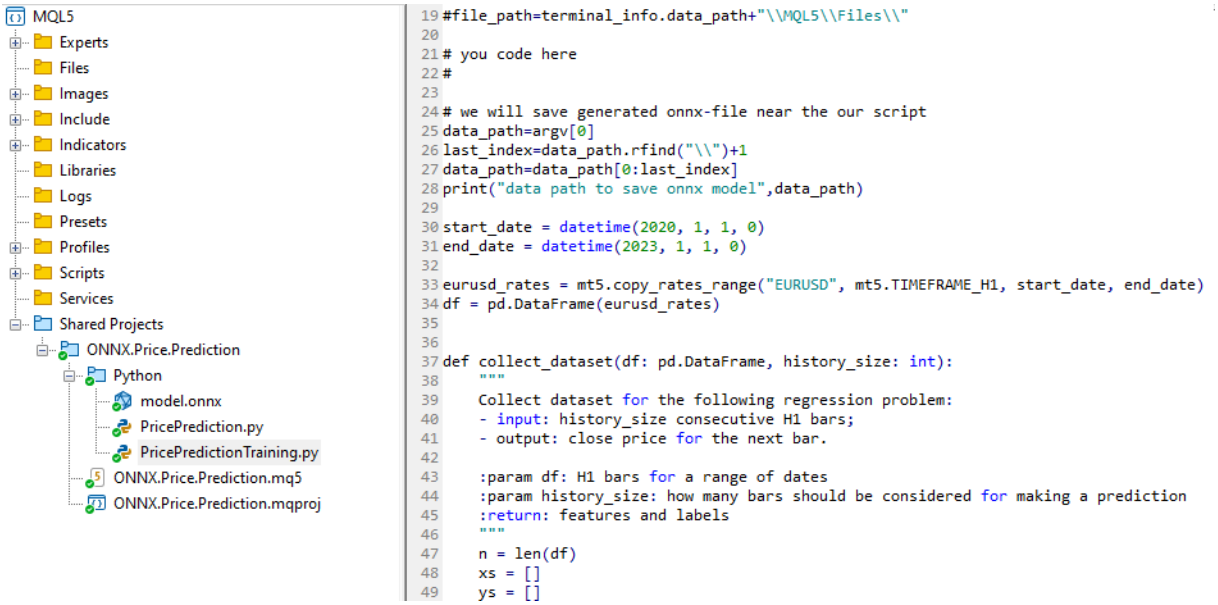
Далее обновите проект из MQL5 Storage.

© 2000-2025, MetaQuotes Ltd.



После этого вам станет доступен проект, в котором вы найдете готовую ONNX модель, два питоновских скрипта, MQL5 скрипт для работы проекта и файл MQL5 проекта (ONNX.Price.Prediction.mqproj).

© 2000-2025, MetaQuotes Ltd.

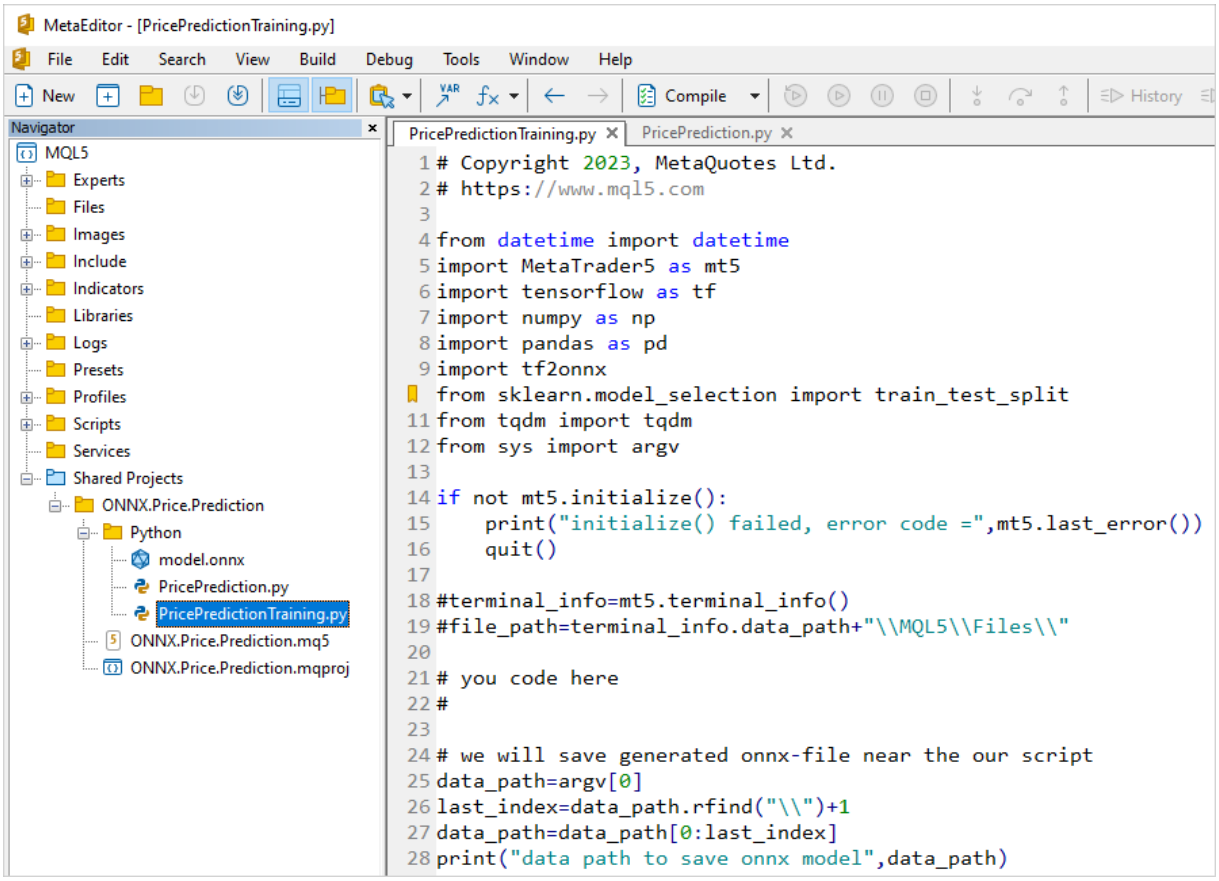


Можно самостоятельно создать ONNX модель с помощью скрипта PricePredictionTraining.py, включенного в проект. Для этого необходимо предварительно из командной строки установить на компьютер необходимые модули.

```
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
```

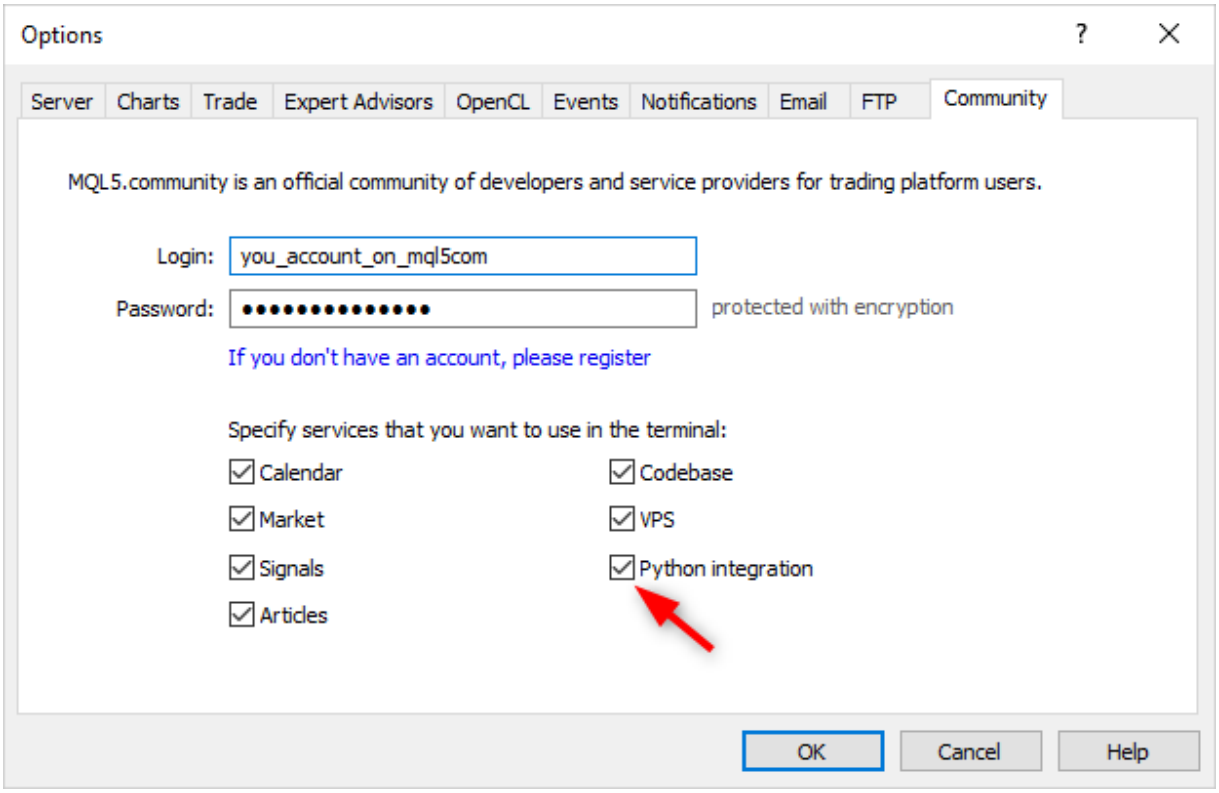
После установки модулей откройте в MetaEditor скрип PricePredictionTraining.py и запустите его на исполнение кнопкой "Компилировать" или клавишей F7.

© 2000-2025, MetaQuotes Ltd.

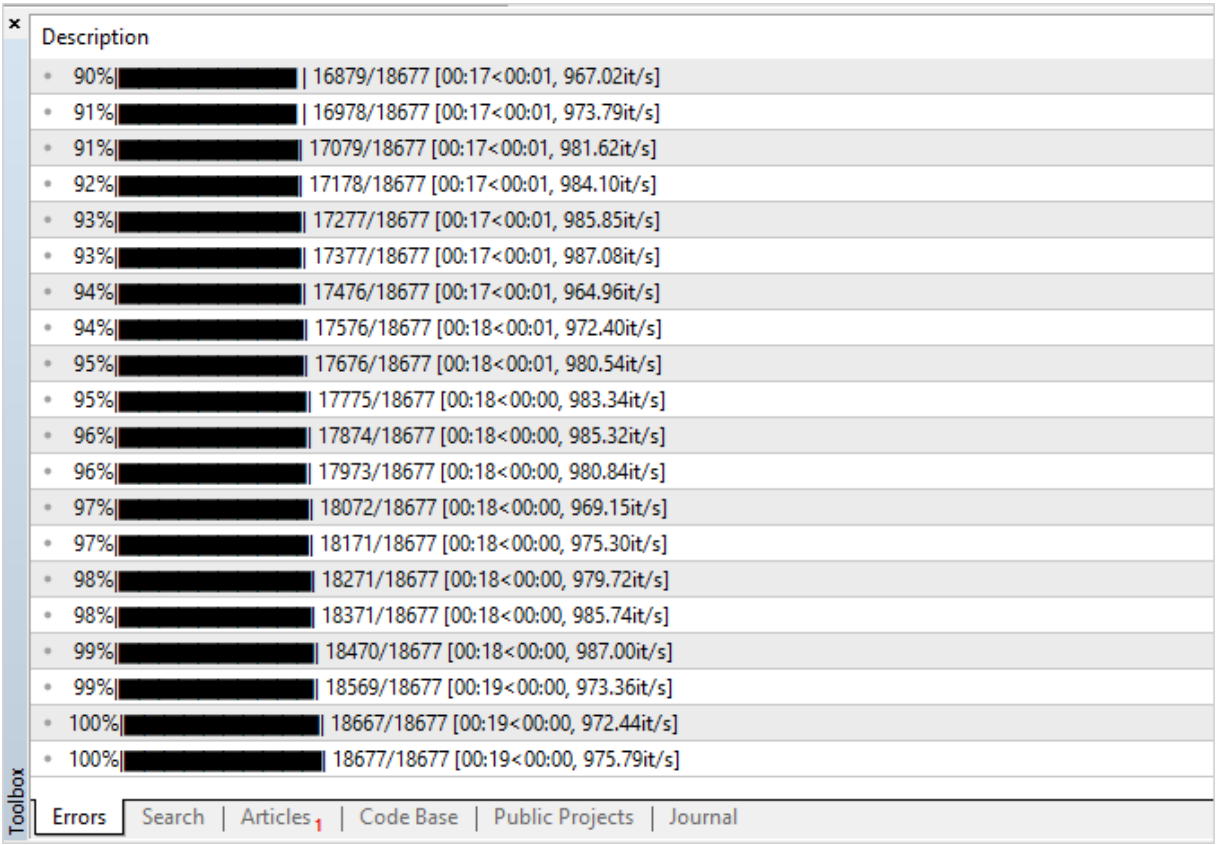


Перед запуском питоновского скрипта убедитесь, что терминал MetaTrader 5 подключен к серверу, на котором есть символ EURUSD. Например, подключитесь к серверу MetaQuotes-Demo и проверьте в настройках терминала чек-бокс "Интеграция с Python"

© 2000-2025, MetaQuotes Ltd.



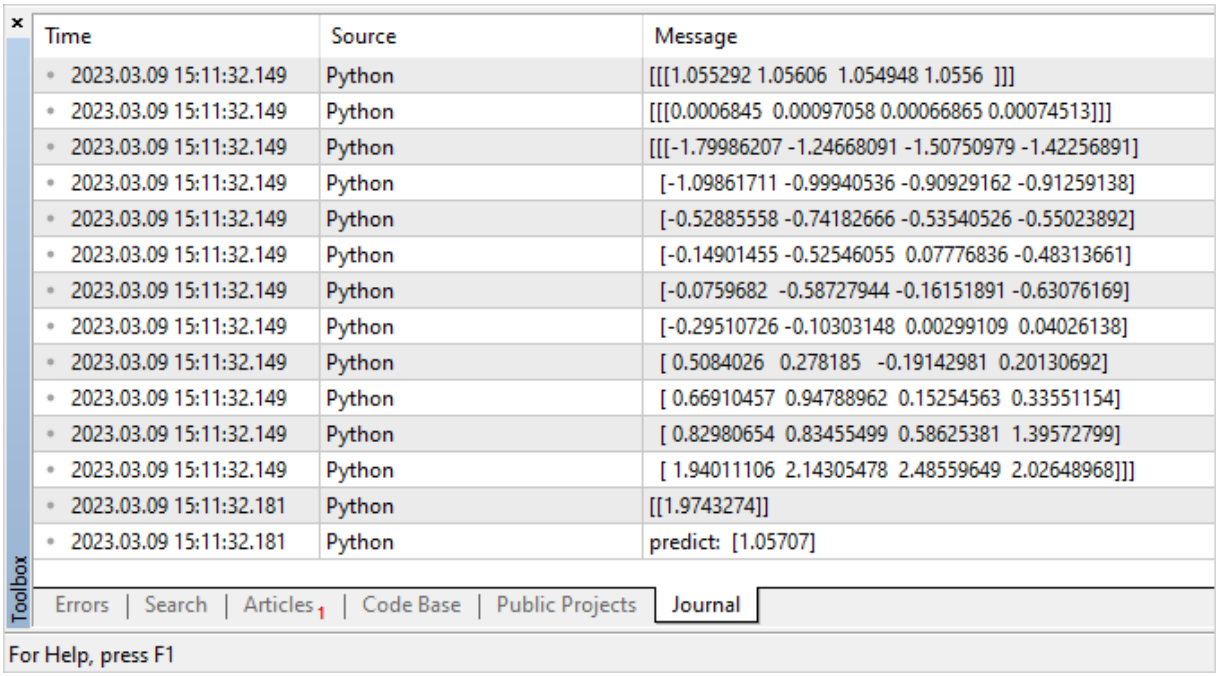
В процессе тренировки сети MetaEditor будет выводить сообщения из питоновского скрипта, пока тренировка не завершится.



© 2000-2025, MetaQuotes Ltd.

При достижении результата в 100% ONNX модель готова и сохраняется в папку проекта <каталог данных терминала>\MQL5\Shared Projects\ONNX.Price.Prediction\Python.

Проверить полученную модель можно запуском второго скрипта PricePrediction.py по кнопке **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]

© 2000-2025, MetaQuotes Ltd.

Запуск модели

Для запуска ONNX модели в MQL5 необходимо выполнить 3 шага:

1. Загрузить её из *.onnx файла с помощью функции OnnxCreate или из массива с помощью OnnxCreateFromBuffer.
2. Указать форму входных и выходных данных функциями OnnxSetInputShape и OnnxSetOutputShape.
3. Запустить модель с помощью OnnxRun, передав ей входные и выходные параметры.
4. Завершить при необходимости работу модели с помощью OnnxRelease.

При создании ONNX модели нужно учитывать существующие лимиты и ограничения, которые описаны в https://github.com/microsoft/onnxruntime/blob/rel-1.14.0/docs/OperatorKernels.md

Например, вот некоторые из них:

Операция	Поддерживаемые типы данных
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)

Ниже представлен пример MQL5 кода из публичного проекта ONNX.Price.Prediction.

```
const long   ExtOutputShape[] = {1,1};     // форма выходных данных модели
const long   ExtInputShape [] = {1,10,4}; // форма входных данных модели
#resource "Python/model.onnx" as uchar ExtModel[]// модель в виде ресурса
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- получаем 10 баров
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- подаем на вход набор векторов 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);
//--- заполним матрицы нормировки
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- нормируем входные данные
   x_norm-=mm;
   x_norm/=ms;
//--- создаём модель
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- укажем форму входных данных
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- укажем форму выходных данных
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- конвертируем нормированные входные данные в тип float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- сюда получим выходные данные модели - предсказание цены
   vectorf y_norm(1);
//--- запускаем модель
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- выведем в лог выходное значение модели
   Print(y_norm);
//--- сделаем обратное преобразование для  получения предсказанной цены
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//--- завершили работу
   OnnxRelease(handle);
   return(0);
  }
```

Пример выполнения скрипта:

```
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
```

© 2000-2025, 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
```

Терминал MetaTrader 5 самостоятельно выбрал оптимального исполнителя для проведения вычислений — ONNX Runtime Execution Provider. В данном случае модель отработала с использованием CPU.

Изменим скрипт, чтобы посчитать процент успешных предсказаний цены Close на основании значений предыдущих 10 баров.

```
#resource "Python/model.onnx" as uchar ExtModel[]// модель в виде ресурса

#define TESTS 10000  // количество тестовых выборок
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- создаём модель
   long session_handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
   if(session_handle==INVALID_HANDLE)
     {
      Print("Cannot create model. Error ",GetLastError());
      return(-1);
     }

//--- так как у модели не определены размеры входного тензора, задаём их явно
//--- первый индекс - размер пачки, второй индекс - размер серии, третий индекс количе
   const long input_shape[]={1,10,4};
   if(!OnnxSetInputShape(session_handle,0,input_shape))
     {
      Print("OnnxSetInputShape error ",GetLastError());
      return(-2);
     }

//--- так как у модели не определены размеры выходного тензора, задаём их явно
//--- первый индекс - размер пачки, должен соответствовать размеру пачки входного тенз
//--- второй индекс - количество предсказанных цен (мы предсказываем только Close)
   const long output_shape[]={1,1};
   if(!OnnxSetOutputShape(session_handle,0,output_shape))
     {
      Print("OnnxSetOutputShape error ",GetLastError());
      return(-3);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- запускаем тесты
   vector closes(TESTS);       // вектор для хранения проверочных цен
   vector predicts(TESTS);     // вектор для хранения полученных предсказаний
   vector prev_closes(TESTS); // вектор для хранения предпоследних цен

   matrix rates;               // матрица для получения серии OHLC
   matrix splitted[2];         // две подматрицы для разделения серии на тестовую и про
   ulong  parts[]={10,1};      // размеры разделяемых подматриц

//--- начинаем с предыдущего бара
   for(int i=1; i<=TESTS; i++)
     {
      //--- получим 11 баров
      rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,i,11);
      //--- делим матрицу на тестовую и проверочную
      rates.Vsplit(parts,splitted);
      //--- из проверочной матрицы берём цену Close
      closes[i-1]=splitted[1][3][0];
      //--- последний Close в тестируемой серии
      prev_closes[i-1]=splitted[0][3][9];

      //--- тестовую матрицу из 10 баров отправляем в тест
      predicts[i-1]=PricePredictionTest(session_handle,splitted[0]);
      //--- ошибка выполнения
      if(predicts[i-1]<=0)
        {
         OnnxRelease(session_handle);
         return(-4);
        }
     }
//--- завершили работу
   OnnxRelease(session_handle);
//--- оценим правильность движения предсказанной цены
   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);
  }
//+------------------------------------------------------------------+
//|  Подготовка данных и запуск модели                               |
//+------------------------------------------------------------------+
double PricePredictionTest(const long session_handle,matrix& rates)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   static matrixf input_data(10,4); // матрица для преобразованных входных данных
   static vectorf output_data(1);   // вектор для получения результата
   static matrix mm(10,4);          // матрица горизонтальных векторов Mean
   static matrix ms(10,4);          // матрица горизонтальных векторов Std

//--- на вход модели должен подаваться набор вертикальных векторов OHLC
   matrix x_norm=rates.Transpose();
//--- нормируем цены
   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;

//--- запускаем модель
   input_data.Assign(x_norm);
   if(!OnnxRun(session_handle,ONNX_DEBUG_LOGS,input_data,output_data))
     {
      Print("OnnxRun error ",GetLastError());
      return(0);
     }
//--- обратно разнормируем цену из выходного значения
   double y_pred=output_data[0]*s[3]+m[3];

   return(y_pred);
  }
```

Запускаем скрипт и получаем точность предсказаний примерно 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 %
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Проверка модели в тестере стратегий

Модели для работы с финансовыми рынками можно проверять в тестере стратегий терминала MetaTrader 5. Это самый быстрый и удобный вариант, не требующий дополнительных усилий по эмуляции рыночного окружения и торговых условий.

Для проверки модели перепишем код из публичного проекта ONNX.Price.Prediction в советника. Это потребует небольших правок.

Создание модели перенесем в функцию OnInit, а завершение onnx-сессии в OnDeinit. Основной блок по работе с моделью поместим в обработчик OnTick.

Добавим также получение цены закрытия 2-х предыдущих баров, чтобы сравнивать актуальную цену закрытия и предсказание.

Код советника небольшой и читается просто.

```
const long   ExtInputShape [] = {1,10,4}; // форма входных данных модели
const long   ExtOutputShape[] = {1,1};     // форма выходных данных модели
#resource "Python/model.onnx" as uchar ExtModel[];// модель в виде ресурса

long handle;          // хендл модели
ulong predictions=0; // счетчик предсказаний
ulong confirmed=0;   // счетчик успешных предсказаний
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- базовые проверки
   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);
     }
//--- создаём модель
   handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- укажем форму входных данных
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

© 2000-2025, MetaQuotes Ltd.

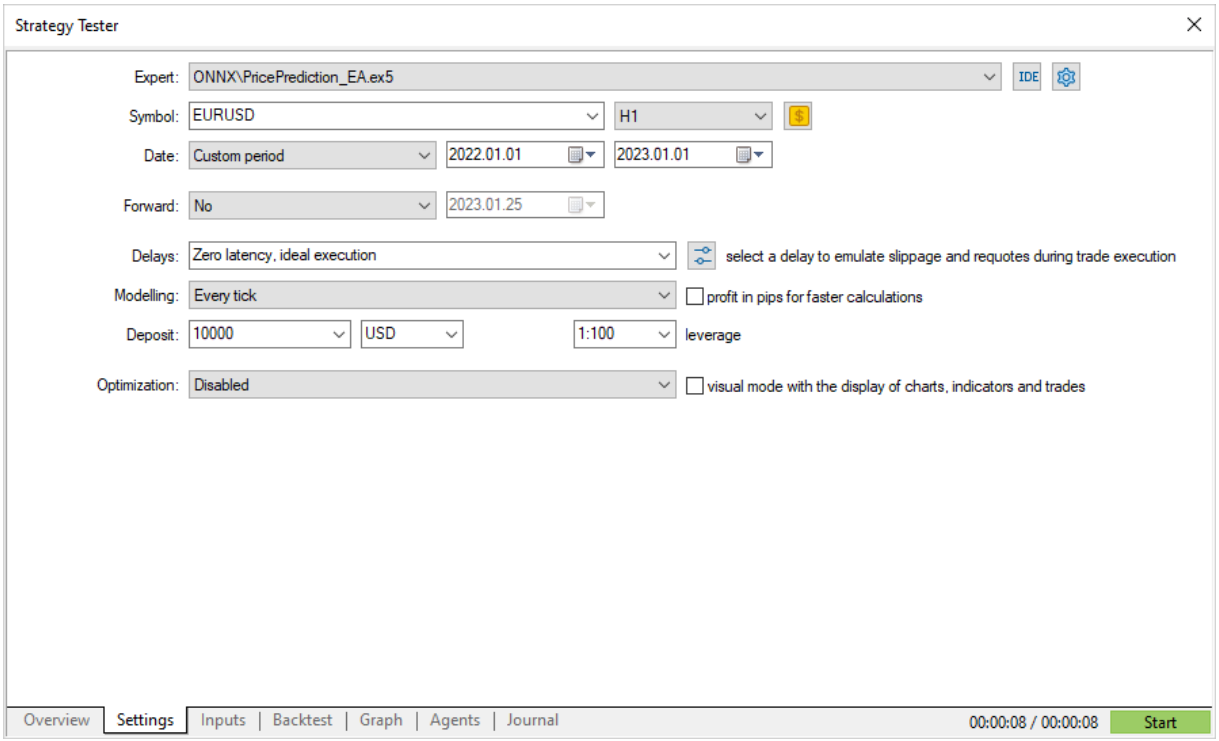
```
//--- укажем форму выходных данных
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- завершили работу модели
   OnnxRelease(handle);
//--- посчитаем и выведем статистику предсказаний
   PrintFormat("Successfull predictions = %.2f %%",confirmed*100./double(predictions))
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static datetime open_time=0;
   static double predict;
//--- проверим время открытия текущего бара
   datetime time=iTime(_Symbol,_Period,0);
   if(time==0)
     {
      PrintFormat("Failed to get Time(0), error %d", GetLastError());
      return;
     }
//--- если время открытия бара не изменилось, то выходим до следующего вызова OnTick
   if(time==open_time)
      return;
//--- получим цены закрытия 2-х последних завершенных баров
   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]; // предсказанное изменение цены
   double delta_actual=close[1]-close[0]; // действительное изменение цены
   if((delta_predict>0 && delta_actual>0) || (delta_predict<0 && delta_actual<0))
      confirmed++;
```

© 2000-2025, MetaQuotes Ltd.

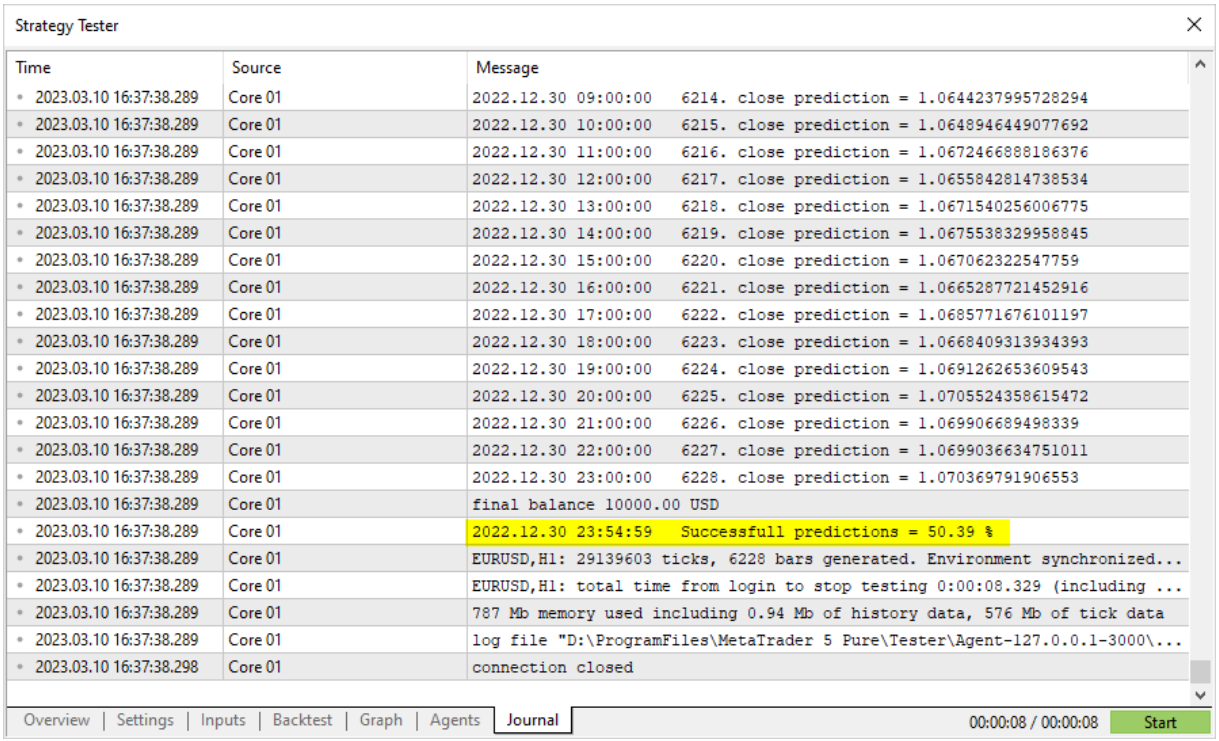
```
//--- вычислим на новом баре цену закрытия, чтобы проверить на следующем баре
   matrix rates;
//--- получаем 10 баров
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,1,10))
      return;
//--- подаем на вход набор векторов OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- заполним матрицы нормировки
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- нормируем входные данные
   x_norm-=mm;
   x_norm/=ms;
//--- конвертируем нормированные входные данные в тип float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- сюда получим выходные данные модели - предсказание цены
   vectorf y_norm(1);
//--- запускаем модель
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
     }
//--- сделаем обратное преобразование для получения предсказанной цены и проверки на
   predict=y_norm[0]*s[3]+m[3];
   predictions++;  // увеличим счетчик предсказаний
   Print(predictions,". close prediction = ",predict);
//--- запомним время открытия бара для проверки на следующем тике
   open_time=time;
  }
```

Компилируем советника и запускаем тестирование на интервале 2022 года, указывем символ EURUSD и таймфрейм H1, на которых тренировалась модель. Режим моделирования тиков значения не имеет, так как в коде есть проверка появления нового бара.

© 2000-2025, MetaQuotes Ltd.



Запускаем и получаем результат в журнале тестирования — чуть более 50% верных предсказаний за 2022 год.



В случае, если результаты предварительной проверки модели удовлетворительны, можно приступать к написанию полноценной торговой стратегии с её использованием.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

OnnxCreate

Создает сессию ONNX с загрузкой модели из *.onnx файла.

```
long  OnnxCreate(
   string  filename,  // путь к файлу
   uint    flags      // флаги для создания модели
   );
```

Параметры

filename

[in] Путь к *.onnx файлу модели относительно папки \MQL5\Files\.

flags

[in] Флаги из ENUM_ONNX_FLAGS, описывающие режим создания модели — ONNX_COMMON_FOLDER и ONNX_DEBUG_LOGS.

Возвращаемое значение

Хендл созданной сессии или INVALID_HANDLE в случае ошибки. Для получения кода ошибки вызовите функцию GetLastError.

Примечание

Если указанный файл не найден на диске, предпринимается повторная попытка открыть файл с добавлением к имени расширения '.onnx'.

© 2000-2025, MetaQuotes Ltd.

OnnxCreateFromBuffer

Создает сессию ONNX с загрузкой модели из массива данных.

```
long  OnnxCreateFromBuffer(
   const uchar&  buffer[],    // ссылка на массив
   ulong         flags        // флаги для создания модели
   );
```

Параметры

buffer

[in] Массив с данными ONNX модели.

flags

[in] Флаги из ENUM_ONNX_FLAGS, описывающие режим создания модели — ONNX_COMMON_FOLDER и ONNX_DEBUG_LOGS.

Возвращаемое значение

Хендл созданной сессии или INVALID_HANDLE в случае ошибки. Для получения кода ошибки вызовите функцию GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxRelease

Удаление сессии ONNX.

```
bool  OnnxRelease(
   long   onnx_handle  // хэндл сессии ONNX
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки вызовите функцию GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxRun

Запуск модели ONNX на исполнение.

```
bool  OnnxRun(
   long     onnx_handle,  // хэндл сессии ONNX
   ulong    flags,        // флаги, описывающие режим запуска
   ...                    // входные и выходные параметры модели
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

flags

[in] Флаги из ENUM_ONNX_FLAGS, описывающие режим запуска — ONNX_DEBUG_LOGS и ONNX_NO_CONVERSION.

...

[in] [out] Входные и выходные параметры модели.

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки вызовите функцию GetLastError.

ENUM_ONNX_FLAGS

Идентификатор	Описание
ONNX_DEBUG_LOGS	Вывод логов отладки
ONNX_NO_CONVERSION	Запрет на автоконверсию, использовать данные пользователя как есть
ONNX_COMMON_FOLDER	Загрузка файла модели из папки Common\Files, по значению равен флагу FILE_COMMON

Пример:

```
const long                              ExtOutputShape[] = {1,1};     // форма выходных
const long                              ExtInputShape [] = {1,10,4}; // форма входных д
#resource "Python/model.onnx" as uchar ExtModel[]                    // модель в виде р
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- получаем 10 баров
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- подаем на вход набор векторов 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);
//--- заполним матрицы нормировки
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- нормируем входные данные
   x_norm-=mm;
   x_norm/=ms;
//--- создаём модель
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- укажем форму входных данных
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- укажем форму выходных данных
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- конвертируем нормированные входные данные в тип float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- сюда получим выходные данные модели - предсказание цены
   vectorf y_norm(1);
//--- запускаем модель
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- выведем в лог выходное значение модели
   Print(y_norm);
//--- сделаем обратное преобразование для  получения предсказанной цены
   double y_pred=y_norm[0]*s[3]+m[3];
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("price predicted:",y_pred);
//--- завершили работу
   OnnxRelease(handle);
   return(0);
  };
```

Смотри также

OnnxSetInputShape, OnnxSetOutputShape

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputCount

Получение количества входных параметров модели ONNX.

```
long  OnnxGetInputCount(
   long   onnx_handle  // хэндл сессии ONNX
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

Возвращаемое значение

В случае успешного выполнения возвращает количество входных параметров, иначе -1. Для получения кода ошибки вызовите функцию GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputCount

Получение количества выходных параметров модели ONNX.

```
long  OnnxGetOutputCount(
   long   onnx_handle  // хэндл сессии ONNX
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

Возвращаемое значение

В случае успешного выполнения возвращает количество выходных параметров, иначе -1. Для получения кода ошибки вызовите функцию GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputName

Получение имени входного параметра модели по индексу.

```
string  OnnxGetInputName(
   long   onnx_handle,  // хэндл сессии ONNX
   long   index         // индекс параметра
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

index

[in] Индекс входного параметра, начиная с 0.

Возвращаемое значение

В случае успешного выполнения возвращает имя входного параметра, иначе NULL. Для получения кода ошибки вызовите функцию GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputName

Получение имени выходного параметра модели по индексу.

```
string  OnnxGetOutputName(
   long   onnx_handle,  // хэндл сессии ONNX
   long   index         // индекс параметра
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

index

[in] Индекс выходного параметра, начиная с 0.

Возвращаемое значение

В случае успешного выполнения возвращает имя выходного параметра, иначе NULL. Для получения кода ошибки вызовите функцию GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputTypeInfo

Получение описания типа входного параметра у модели.

```
bool  OnnxGetInputTypeInfo(
   long           onnx_handle,  // хэндл сессии ONNX
   long           index,        // индекс параметра
   OnnxTypeInfo&  typeinfo      // описание типа параметра
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

index

[in] Индекс входного параметра, начиная с 0.

typeinfo

[out] Структура OnnxTypeInfo, описывающая тип входного параметра.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки вызовите функцию GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputTypeInfo

Получение описания типа выходного параметра у модели.

```
bool  OnnxGetOutputTypeInfo(
   long           onnx_handle,  // хэндл сессии ONNX
   long           index,        // индекс параметра
   OnnxTypeInfo&  typeinfo      // описание типа параметра
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

index

[in] Индекс выходного параметра, начиная с 0.

typeinfo

[out] Структура OnnxTypeInfo, описывающая тип выходного параметра.

Возвращаемое значение

В случае успешного выполнения возвращает true, иначе false. Для получения кода ошибки вызовите функцию GetLastError.

© 2000-2025, MetaQuotes Ltd.

OnnxSetInputShape

Устанавливает размерность входных данных модели по индексу.

```
bool  OnnxSetInputShape(
   long           onnx_handle,   // хэндл сессии ONNX
   long           input_index,   // индекс входного параметра
   const ulong&   shape[]        // массив, описывающий размерность входных данных
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

input_index

[in] Индекс входного параметра, начиная с 0.

shape

[in] Массив, описывающий размерность входных данных модели.

Возвращаемое значение

В случае успешного выполнения возвращает имя входного параметра, иначе NULL. Для получения кода ошибки вызовите функцию GetLastError.

Пример:

```
//---- опишем размерности входных и выходных данных модели
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- создаём модель
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- укажем форму входных данных
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- укажем форму выходных данных
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Смотри также

OnnxSetOutputShape

© 2000-2025, MetaQuotes Ltd.

OnnxSetOutputShape

Устанавливает размерность выходных данных модели по индексу.

```
bool  OnnxSetOutputShape(
   long           onnx_handle,    // хэндл сессии ONNX
   long           output_index,   // индекс выходного параметра
   const ulong&   shape[]         // массив, описывающий размерность выходных данных
   );
```

Параметры

onnx_handle

[in] Хэндл объекта сессии ONNX, созданный через OnnxCreate или OnnxCreateFromBuffer.

output_index

[in] Индекс выходного параметра, начиная с 0.

shape

[in] Массив, описывающий размерность выходных данных модели.

Возвращаемое значение

В случае успешного выполнения возвращает имя входного параметра, иначе NULL. Для получения кода ошибки вызовите функцию GetLastError.

Пример:

```
//---- опишем размерности входных и выходных данных модели
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- создаём модель
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- укажем форму входных данных
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- укажем форму выходных данных
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Смотри также

OnnxSetInputShape

© 2000-2025, MetaQuotes Ltd.

Структуры данных

Для работы с моделями ONNX используются следующие структуры данных:

OnnxTypeInfo

Структура описывает тип входного или выходного параметра модели ONNX

```
struct OnnxTypeInfo
  {
   ENUM_ONNX_TYPE          type;            // тип параметра
   OnnxTensorTypeInfo      tensor;          // описание тензора
   OnnxMapTypeInfo         map;             // описание map
   OnnxSequenceTypeInfo    sequence;        // описание последовательности
  };
```

В качестве входного параметра может использоваться только тензор (ONNX_TYPE_TENSOR), в этом случае заполняется значениями только поле OnnxTypeInfo::tensor, остальные поля (map и sequence) не определены.

В качестве выходного параметра может использоваться только один из трех типов OnnxTypeInfo (ONNX_TYPE_TENSOR, ONNX_TYPE_MAP или ONNX_TYPE_SEQUENCE), в зависимости от типа заполняется соответствующая подструктура (OnnxTypeInfo::tensor, OnnxTypeInfo::map или OnnxTypeInfo::sequence).

OnnxTensorTypeInfo

Структура описывает тензор во входном или выходном параметре модели ONNX

```
struct OnnxTensorTypeInfo
  {
   const ENUM_ONNX_DATA_TYPE    data_type;       // тип данных в тензоре
   const long                   dimensions[];    // количество элементов в тензоре
  };
```

OnnxMapTypeInfo

Структура описывает map, который получается в выходном параметре модели ONNX

```
struct OnnxMapTypeInfo
  {
   const ENUM_ONNX_DATA_TYPE    key_type;        // тип ключа
   const OnnxTypeInfo&          value_type;      // тип значения
  };
```

OnnxSequenceTypeInfo

© 2000-2025, MetaQuotes Ltd.

Структура описывает последовательность, которая получается в выходном параметре модели ONNX

```
struct OnnxSequenceTypeInfo
  {
   const OnnxTypeInfo&          value_type;      // тип данных в последовательности
  };
```

ENUM_ONNX_TYPE

Перечисление **ENUM_ONNX_TYPE** описывает тип параметра модели

Идентификатор	Описание
ONNX_TYPE_UNKNOWN	Неизвестен
ONNX_TYPE_TENSOR	Тензор
ONNX_TYPE_SEQUENCE	Последовательность
ONNX_TYPE_MAP	Карта
ONNX_TYPE_OPAQUE	Абстрактный
ONNX_TYPE_SPARSETENSOR	Разреженный тензор

ENUM_ONNX_DATA_TYPE

Перечисление **ENUM_ONNX_DATA_TYPE** описывает тип используемых данных

Идентификатор	Описание
ONNX_DATA_TYPE_UNDEFINED	Не определен
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_INT8	8-битный int
ONNX_DATA_TYPE_UINT16	16-битный uint
ONNX_DATA_TYPE_INT16	16-битный int
ONNX_DATA_TYPE_INT32	32-битный int
ONNX_DATA_TYPE_INT64	64-битный int
ONNX_DATA_TYPE_STRING	string
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT16	16-битный float
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	32-битный uint

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
ONNX_DATA_TYPE_UINT64	64-битный uint
ONNX_DATA_TYPE_COMPLEX64	64-битное комплексное число
ONNX_DATA_TYPE_COMPLEX128	128-битное комплексное число
ONNX_DATA_TYPE_BFLOAT16	16-битный bfloat (Brain Floating Point)

ENUM_ONNX_FLAGS

Перечисление **ENUM_ONNX_FLAGS** описывает режим запуска модели

Идентификатор	Описание
ONNX_DEBUG_LOGS	Вывод логов отладки
ONNX_NO_CONVERSION	Запрет на автоконверсию, использовать данные пользователя как есть
ONNX_COMMON_FOLDER	Загрузка файла модели из папки Common\Files, по значению равен флагу FILE_COMMON

Конвертация массивов при работе с ONNX моделями

В задачах машинного обучения не всегда нужна большая точность вычислений. Для ускорения расчетов некоторые модели используют типы данных с меньшей точностью, такие как Float16 и даже Float8. Чтобы пользователи могли подавать такие данные на вход моделей, в MQL5 добавлены 4 специальные функции, которые преобразует стандартные типы MQL5 в специальные типы FP16 и FP8.

Функция	Действие
ArrayToFP16	Производит копирование массива типа float или double в массив типа ushort с заданным форматом
ArrayToFP8	Производит копирование массива типа float или double в массив типа uchar с заданным форматом
ArrayFromFP16	Производит копирование массива типа ushort в массив типа float или double с заданным форматом
ArrayFromFP8	Производит копирование массива типа uchar в массив типа float или double с заданным форматом

Данные функции для преобразования массивов используют специальные форматы, заданные в перечислениях ниже.

ENUM_FLOAT16_FORMAT

© 2000-2025, MetaQuotes Ltd.

Перечисление **ENUM_FLOAT16_FORMAT** описывает 2 формата типа FP16.

Идентификатор	Описание
FLOAT_FP16	Стандартный 16-битный формат, так же известный как half
FLOAT_BFP16	Специальный формат brain float point

Оба этих формата имеют свои преимущества и ограничения. FLOAT16 обеспечивает более высокую точность, но требует больше ресурсов для хранения и вычислений. BFLOAT16, с другой стороны, обеспечивает более высокую производительность и эффективность при обработке данных, но может быть менее точным.

ENUM_FLOAT8_FORMAT

Перечисление **ENUM_FLOAT8_FORMAT** описывает 4 формата типа FP8.

Формат FP8 (8-битное число с плавающей запятой) представляет собой один из типов данных, используемых для представления чисел с плавающей точкой. В FP8 каждое число представлено 8 битами данных, которые обычно разделяются на три компонента: знак, экспоненту и мантиссу. Этот формат обеспечивает компромисс между точностью и эффективностью хранения данных, что делает его привлекательным для использования в приложениях, где требуется экономия памяти и вычислительных ресурсов.

Идентификатор	Описание
FLOAT_FP8_E4M3FN	8-битное число с плавающей точкой, 4 бита порядок и 3 бита мантисса. Обычно используется как коэффициенты.
FLOAT_FP8_E4M3FNUZ	8-битное число с плавающей точкой, 4 бит порядок и 3 бита мантисса. Поддерживает NaN, не поддерживается отрицательный ноль и Inf. Обычно используется как коэффициенты.
FLOAT_FP8_E5M2FN	8-битное число с плавающей точкой, 5 бит порядок и 2 бита мантисса. Поддерживает NaN и Inf. Обычно используется для градиентов.
FLOAT_FP8_E5M2FNUZ	8-битное число с плавающей точкой, 5 бит порядок и 2 бита мантисса. Поддерживает NaN и Inf, не поддерживает отрицательный ноль. Также используется для градиентов.

Одним из ключевых преимуществ FP8 является его эффективность при обработке больших объемов данных. Благодаря компактному представлению чисел, FP8 позволяет уменьшить требования к памяти и ускорить вычисления. Это особенно важно в приложениях машинного обучения и искусственного интеллекта, где обработка больших наборов данных является обычным делом.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Стандартная библиотека

Эта группа разделов содержит технические детали работы со стандартной библиотекой MQL5 и описания всех ее ключевых компонентов.

Стандартная библиотека MQL5 написана на языке MQL5 и предназначена для облегчения написания программ (индикаторов, скриптов, экспертов) конечным пользователям. Библиотека обеспечивает удобный доступ к большинству внутренних функций MQL5.

Стандартная библиотека MQL5 размещается в рабочем каталоге терминала в папке Include.

Раздел	Размещение
Математика	Include\Math\
OpenCL	Include\OpenCL\
Базовый класс CObject	Include\
Коллекции данных	Include\Arrays\
Шаблонные коллекции данных	Include\Generic\
Файлы	Include\Files\
Строки	Include\Strings\
Графические объекты	Include\Objects\
Пользовательская графика	Include\Canvas\
3D графика	Include\Canvas\
Ценовые графики	Include\Charts\
Научные графики	Include\Graphics\
Индикаторы	Include\Indicators\
Торговые классы	Include\Trade\
Модули стратегий	Include\Expert\
Панели и диалоги	Include\Controls\

© 2000-2025, MetaQuotes Ltd.

Математика

Для проведения вычислений в разных областях математики предлагается несколько библиотек:

- Статистика – функции для работы с различными распределениями из теории вероятности
- Нечеткая логика – библиотека нечеткой логики, в которой реализованы системы нечеткого логического вывода Мамдани и Сугено
- ALGLIB – анализ данных (кластеризация, деревья решений, линейная регрессия, нейронные сети), решения дифференциальных уравнений, преобразования Фурье, численное интегрирование, задачи оптимизации, статистический анализа и многое другое.

© 2000-2025, MetaQuotes Ltd.

Статистика

Статистическая библиотека предназначена для удобной работы с основными статистическими распределениями.

Для каждого распределения в библиотеке представлено 5 функций:

1. Расчет плотности распределения – функции вида MathProbabilityDensityX()
2. Расчет вероятностей – функции вида MathCumulativeDistributionX()
3. Расчет квантилей распределений – функции вида MathQuantileX()
4. Генерация случайных чисел с заданным распределением – функции вида MathRandomX()
5. Расчет теоретических моментов распределений – функции вида MathMomentsX()

Кроме расчета значений для отдельных случайных величин в библиотеке представлены также перегрузки функций, которые производят такие же расчеты на массивах.

- Статистические характеристики
- Нормальное распределение
- Логнормальное распределение
- Бета-распределение
- Нецентральное бета-распределение
- Гамма-распределение
- Распределение хи-квадрат
- Нецентральное распределение хи-квадрат
- Экспоненциальное распределение
- F-распределение
- Нецентральное F-распределение
- T-распределение
- Нецентральное T-распределение
- Логистическое распределение
- Распределение Коши
- Равномерное распределение
- Распределение Вейбулла
- Биномиальное распределение
- Отрицательное биномиальное распределение
- Геометрическое распределение
- Гипергеометрическое распределение
- Распределение Пуассона
- Вспомогательные функции

Пример:

© 2000-2025, 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"
//--- подключаем функции для расчета нормального распределения
#include <Math\Stat\Normal.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- задаем параметры нормального распределения
   double mu=5.0;
   double sigma=1.0;
   PrintFormat("Нормальное распределение с параметрами mu=%G и sigma=%G, примеры вычис
//--- задаем интервал
   double x1=mu-sigma;
   double x2=mu+sigma;
//--- переменные для расчета вероятности
   double cdf1,cdf2,probability;
//--- переменные для кода ошибки
   int error_code1,error_code2;
//--- рассчитаем значения функции распределения
   cdf1=MathCumulativeDistributionNormal(x1,mu,sigma,error_code1);
   cdf2=MathCumulativeDistributionNormal(x2,mu,sigma,error_code2);
//--- проверим код ошибок
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- рассчитаем вероятность случайной величины в диапазоне
      probability=cdf2-cdf1;
      //--- выводим результат
      PrintFormat("1. Вычислить в диапазоне %.5f<x<%.5f вероятность случайной величины
      PrintFormat("  Ответ: Probability = %5.8f",probability);
     }

//--- Найдем интервал значений случайной величины x, соответствующий 95% доверительной
   probability=0.95; //  задаем доверительную вероятность
//--- задаем вероятности на границах интервала
   double p1=(1.0-probability)*0.5;
   double p2=probability+(1.0-probability)*0.5;
//--- вычислим границы интервала
   x1=MathQuantileNormal(p1,mu,sigma,error_code1);
   x2=MathQuantileNormal(p2,mu,sigma,error_code2);
//--- проверим код ошибок
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- выводим результат
      PrintFormat("2. Для доверительного интервала = %.2f найти дипазон случайной вели
      PrintFormat("  Ответ: диапазон  %5.8f <= x <=%5.8f",x1,x2);
     }

   PrintFormat("3. Вычислить рассчитанные и теоретические 4 первые момента распределен
//--- Сгенерируем массив случайных чисел, рассчитаем первые 4 момента и сравним с теор
   int data_count=1000000;  // задаем количество значений и подготавливаем массив
   double data[];
   ArrayResize(data,data_count);
//--- генерируем случайные значения и сохраняем их в массив
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<data_count; i++)
     {
      data[i]=MathRandomNormal(mu,sigma,error_code1);
     }
//--- задаем индекс начального значения и количество данных для расчета
   int start=0;
   int count=data_count;
//--- вычислим первые 4 момента сгенерированных значений
   double mean=MathMean(data,start,count);
   double variance=MathVariance(data,start,count);
   double skewness=MathSkewness(data,start,count);
   double kurtosis=MathKurtosis(data,start,count);
//--- переменные для теоретических моментов
   double normal_mean=0;
   double normal_variance=0;
   double normal_skewness=0;
   double normal_kurtosis=0;
//--- выводим значения рассчитанных моментов
   PrintFormat("            Mean           Variance           Skewness           Kurtos
   PrintFormat("Calculated  %.10f   %.10f      %.10f      %.10f",mean,variance,skewnes
//--- рассчитываем теоретические значения моментов и сравниваем с полученными
   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,
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Статистические характеристики

Эта группа функций рассчитывает статистические характеристики элементов массива:

- среднее,
- дисперсию,
- коэффициент асимметрии,
- коэффициент эксцесса,
- медиану,
- среднеквадратичное и
- стандартные отклонения.

Функция	Описание
MathMean	Рассчитывает среднее значение (первый момент) элементов массива
MathVariance	Рассчитывает дисперсию (второй момент) элементов массива
MathSkewness	Рассчитывает коэффициент асимметрии (третий момент) элементов массива
MathKurtosis	Рассчитывает коэффициент эксцесса (четвертый момент) элементов массива
MathMoments	Рассчитывает первые 4 момента (среднее, дисперсия, коэффициент асимметрии, коэффициент эксцесса) элементов массива
MathMedian	Рассчитывает медианное значение элементов массива
MathStandardDeviation	Рассчитывает стандартное отклонение элементов массива
MathAverageDeviation	Рассчитывает среднее отклонение элементов массива

© 2000-2025, MetaQuotes Ltd.

MathMean

Рассчитывает среднее значение (первый момент) элементов массива. Аналог mean() в R.

```
double  MathMean(
   const double&  array[]                // массив с данными
   );
```

Параметры

array

[in] Массив с данными для расчета среднего значения.

start=0

[in] Начальный индекс для расчета.

count=WHOLE_ARRAY

[in] Количество элементов для расчета.

Возвращаемое значение

Среднее значение элементов массива. В случае ошибки возвращает NaN (не число).

© 2000-2025, MetaQuotes Ltd.

MathVariance

Рассчитывает дисперсию (второй момент) элементов массива. Аналог var() в R.

```
double  MathVariance(
   const double&  array[]                // массив с данными
   );
```

Параметры

array

[in] Массив с данными для расчета.

start=0

[in] Начальный индекс для расчета.

count=WHOLE_ARRAY

[in] Количество элементов для расчета.

Возвращаемое значение

Дисперсия элементов массива. В случае ошибки возвращает NaN (не число).

© 2000-2025, MetaQuotes Ltd.

MathSkewness

Рассчитывает коэффициент асимметрии (третий момент) элементов массива. Аналог skewness() в R (библиотека e1071).

```
double  MathSkewness(
   const double&  array[]                // массив с данными
   );
```

Параметры

array

[in] Массив с данными для расчета.

start=0

[in] Начальный индекс для расчета.

count=WHOLE_ARRAY

[in] Количество элементов для расчета.

Возвращаемое значение

Коэффициент асимметрии элементов массива. В случае ошибки возвращает NaN (не число).

© 2000-2025, MetaQuotes Ltd.

MathKurtosis

Рассчитывает коэффициент эксцесса (четвертый момент) элементов массива. Аналог kurtosis() в R (библиотека e1071).

```
double  MathKurtosis(
   const double&  array[]                // массив с данными
   );
```

Параметры

array

[in] Массив с данными для расчета.

start=0

[in] Начальный индекс для расчета.

count=WHOLE_ARRAY

[in] Количество элементов для расчета.

Возвращаемое значение

Коэффициент эксцесса элементов массива. В случае ошибки возвращает NaN (не число).

Примечание

Расчет коэффициента эксцесса производится относительно нормального распределения (excess kurtosis=kurtosis-3), т.е. коэффициент эксцесса нормального распределения равен нулю.

Он положителен, если пик распределения около математического ожидания острый, и отрицателен, если вершина гладкая.

© 2000-2025, MetaQuotes Ltd.

MathMoments

Рассчитывает первые 4 момента (среднее, дисперсия, коэффициент асимметрии, коэффициент эксцесса) элементов массива.

```
double  MathMoments(
   const double&  array[],               // массив с данными
   double&        mean,                  // среднее значение (1-ый момент)
   double&        variance,              // дисперсия (2-ой момент)
   double&        skewness,              // коэффициент асимметрии (3-ий момент)
   double&        kurtosis,              // коэффициент эксцесса (4-ый момент)
   const int      start=0,               // начальный индекс
   const int      count=WHOLE_ARRAY      // количество элементов
   );
```

Параметры

array

[in] Массив с данными для расчета.

mean

[out] Переменная для среднего значения (1 момент)

variance

[out] Переменная для дисперсии (2 момент)

skewness

[out] Переменная для коэффициента асимметрии (3 момент)

kurtosis

[out] Переменная для коэффициента эксцесса (4 момент)

start=0

[in] Начальный индекс для расчета.

count=WHOLE_ARRAY

[in] Количество элементов для расчета.

Возвращаемое значение

Возвращает true, если моменты успешно рассчитаны, иначе false.

Примечание

Расчет коэффициента эксцесса производится относительно нормального распределения (excess kurtosis=kurtosis-3), т.е. коэффициент эксцесса нормального распределения равен нулю.

Он положителен, если пик распределения около математического ожидания острый, и отрицателен, если вершина гладкая.

© 2000-2025, MetaQuotes Ltd.

MathMedian

Рассчитывает медианное значение элементов массива. Аналог median() в R.

```
double  MathMedian(
   const double&  array[]                // массив с данными
   );
```

Параметры

array

[in] Массив с данными для расчета.

start=0

[in] Начальный индекс для расчета.

count=WHOLE_ARRAY

[in] Количество элементов для расчета.

Возвращаемое значение

Медианное значение элементов массива. В случае ошибки возвращает NaN (не число).

© 2000-2025, MetaQuotes Ltd.

MathStandardDeviation

Рассчитывает стандартное отклонение элементов массива. Аналог sd() в R.

```
double  MathStandardDeviation(
   const double&  array[]                // массив с данными
   );
```

Параметры

array

[in] Массив с данными для расчета.

start=0

[in] Начальный индекс для расчета.

count=WHOLE_ARRAY

[in] Количество элементов для расчета.

Возвращаемое значение

Стандартное отклонение элементов массива. В случае ошибки возвращает NaN (не число).

© 2000-2025, MetaQuotes Ltd.

MathAverageDeviation

Рассчитывает среднее отклонение элементов массива. Аналог aad() в R.

```
double  MathAverageDeviation(
   const double&  array[]                // массив с данными
   );
```

Параметры

array

[in] Массив с данными для расчета.

start=0

[in] Начальный индекс для расчета.

count=WHOLE_ARRAY

[in] Количество элементов для расчета.

Возвращаемое значение

Среднее отклонение элементов массива. В случае ошибки возвращает NaN (не число).

© 2000-2025, MetaQuotes Ltd.

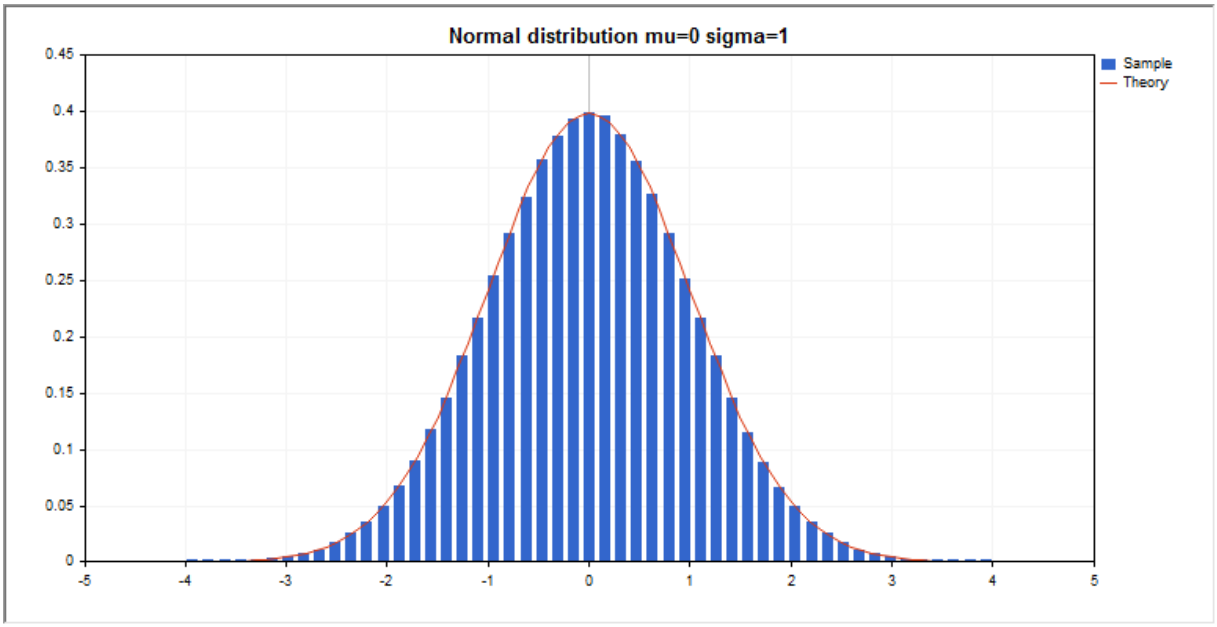
Нормальное распределение

В данном разделе представлены функции для работы с нормальным распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по нормальному закону. Распределение описывается следующей формулой:

$$f_{Normal}(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

где:

- x — значение случайной величины
- μ — математическое ожидание
- σ — среднеквадратическое отклонение



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityNormal	Рассчитывает плотность вероятности нормального распределения
MathCumulativeDistributionNormal	Рассчитывает значение функции нормального распределения вероятностей
MathQuantileNormal	Рассчитывает значение обратной функции нормального распределения для заданной вероятности
MathRandomNormal	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по нормальному закону

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathMomentsNormal	Рассчитывает теоретические численные значения первых 4 моментов нормального распределения

Пример:

```
#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;  // математическое ожидание (mean)
input double std_dev=1;     // среднеквадратическое отклонение (standard deviation)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // количество значений в выборке
   int ncells=51;        // количество интервалов в гистограмме
   double x[];           // центры интервалов гистограммы
   double y[];           // количество значений из выборки, попавших в интервал
   double data[];        // выборка случайных значений
   double max,min;       // максимальное и минимальное значения в выборке
//--- получим выборку из нормального распределения
   MathRandomNormal(mean_value,std_dev,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNormal(x2,mean_value,std_dev,false,y2);
//--- масштабируем
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNormal

Рассчитывает плотность вероятности нормального распределения с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNormal(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения mean (математическое ожидан
   const double  sigma,          // параметр распределения sigma (среднеквадратическое
   const bool    log_mode,       // расчет логарифма значения
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности нормального распределения с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNormal(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения mean (математическое ожидан
   const double  sigma,          // параметр распределения sigma (среднеквадратическое
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности нормального распределения с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dnorm() в R.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],           // массив со значениями случайной величины
   const double   mu,            // параметр распределения mean (математическое ожида
   const double   sigma,         // параметр распределения sigma (среднеквадратическ
   const bool     log_mode,      // расчет логарифма значения
   double&        result[]       // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности нормального распределения с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],           // массив со значениями случайной величины
   const double   mu,            // параметр распределения mean (математическое ожида
   const double   sigma,         // параметр распределения sigma (среднеквадратическ
   double&        result[]       // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

mu

[in] Параметр распределения mean (математическое ожидание).

sigma

[in] Параметр распределения sigma (среднеквадратическое отклонение).

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив для получения значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNormal

Рассчитывает значение функции нормального распределения вероятностей с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // значение случайной величины
   const double  mu,             // математическое ожидание
   const double  sigma,          // среднеквадратическое отклонение
   const bool    tail,           // флаг расчета хвоста (tail)
   const bool    log_mode,       // расчет логарифма значения
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции нормального распределения вероятностей с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // значение случайной величины
   const double  mu,             // математическое ожидание
   const double  sigma,          // среднеквадратическое отклонение
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции нормального распределения вероятностей с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dnorm() в R.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // математическое ожидание
   const double  sigma,          // среднеквадратическое отклонение
   const bool    tail,           // флаг расчета хвоста (tail)
   const bool    log_mode,       // расчет логарифма значения
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает значение функции нормального распределения вероятностей с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // математическое ожидание
   const double  sigma,          // среднеквадратическое отклонение
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

mu

[in] Параметр распределения mean (математическое ожидание).

sigma

[in] Параметр распределения sigma (среднеквадратическое отклонение).

tail

[in] Флаг расчета. Если tail=true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив для получения значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNormal

Рассчитывает для заданной вероятности *probability* значение обратной функции нормального распределения с параметрами mu и sigma. В случае ошибки возвращает NaN.

```
double  MathQuantileNormal(
   const double  probability,    // значение вероятности случайной величины
   const double  mu,             // математическое ожидание
   const double  sigma,          // среднеквадратическое отклонение
   const bool    tail,           // флаг расчета хвоста (tail)
   const bool    log_mode,       // расчет логарифма значения
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает для заданной вероятности *probability* значение обратной функции нормального распределения с параметрами mu и sigma. В случае ошибки возвращает NaN.

```
double  MathQuantileNormal(
   const double  probability,    // значение вероятности случайной величины
   const double  mu,             // математическое ожидание
   const double  sigma,          // среднеквадратическое отклонение
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает для массива значений вероятности *probability[]* значения обратной функции нормального распределения с параметрами mu и sigma. В случае ошибки возвращает false. Аналог qnorm() в R.

```
bool  MathQuantileNormal(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  mu,             // математическое ожидание
   const double  sigma,          // среднеквадратическое отклонение
   const bool    tail,           // флаг расчета хвоста (tail)
   const bool    log_mode,       // расчет логарифма значения
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает для массива значений вероятности *probability[]* значения обратной функции нормального распределения с параметрами mu и sigma. В случае ошибки возвращает false.

```
bool  MathQuantileNormal(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  mu,             // математическое ожидание
   const double  sigma,          // среднеквадратическое отклонение
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Массив со значениями вероятностей случайной величины.

mu

[in] Параметр распределения mean (математическое ожидание).

sigma

[in] Параметр распределения sigma (среднеквадратическое отклонение).

tail

[in] Флаг расчета. Если false, то расчет ведется для вероятности 1.0 - probability

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив для получения квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomNormal

Генерирует псевдослучайную величину, распределенную по нормальному закону с параметрами mu и sigma. В случае ошибки возвращает NaN.

```
double  MathRandomNormal(
   const double  mu,              // математическое ожидание
   const double  sigma,           // среднеквадратическое отклонение
   int&          error_code       // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по нормальному закону с параметрами mu и sigma. В случае ошибки возвращает false. Аналог rnorm() в R.

```
bool  MathRandomNormal(
   const double  mu,              // математическое ожидание
   const double  sigma,           // среднеквадратическое отклонение
   const int     data_count,      // количество необходимых значений
   double&       result[]         // массив для получения псевдослучайных величин
   );
```

Параметры

mu

[in] Параметр распределения mean (математическое ожидание).

sigma

[in] Параметр распределения sigma (среднеквадратическое отклонение).

data_count

[in] Количество псевдослучайных значений, которые необходимо получить.

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNormal

Рассчитывает теоретические численные значения первых 4 моментов нормального распределения.

```
double  MathMomentsNormal(
   const double  mu,              // математическое ожидание
   const double  sigma,           // среднеквадратическое отклонение
   double&       mean,            // переменная для среднего значения
   double&       variance,        // переменная для дисперсии
   double&       skewness,        // переменная для коэффициента асимметрии
   double&       kurtosis,        // переменная для коэффициента эксцесса
   int&          error_code       // переменная для записи кода ошибки
   );
```

Параметры

mu

[in] Параметр распределения mean (математическое ожидание).

sigma

[in] Параметр распределения sigma (среднеквадратическое отклонение).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если моменты успешно рассчитаны, иначе false.

© 2000-2025, MetaQuotes Ltd.

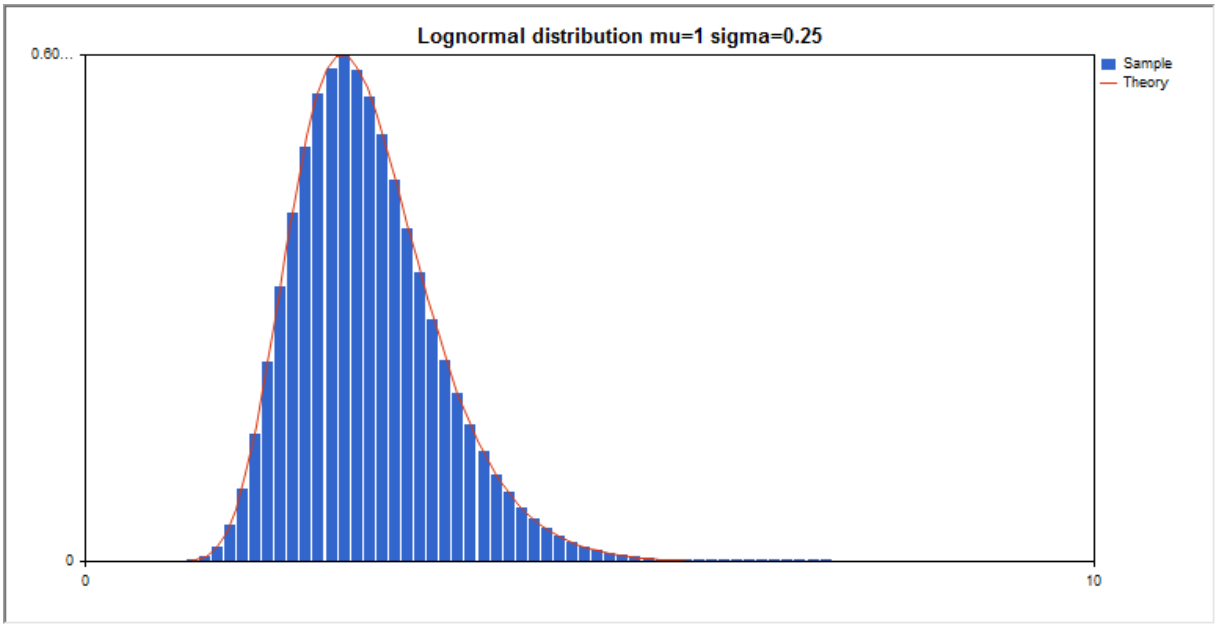
Логнормальное распределение

В данном разделе представлены функции для работы с логнормальным распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по логнормальному закону. Логнормальное распределение описывается следующей формулой:

$$f_{Lognormal}(x|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

где:

- x — значение случайной величины
- μ — логарифм математического ожидания
- σ — логарифм среднеквадратического отклонения



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityLognormal	Рассчитывает плотность вероятности логнормального распределения
MathCumulativeDistributionLognormal	Рассчитывает значение функции логнормального распределения вероятностей
MathQuantileLognormal	Рассчитывает значение обратной функции логнормального распределения для заданной вероятности
MathRandomLognormal	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по логнормальному закону
MathMomentsLognormal	Рассчитывает теоретические численные значения первых 4 моментов логнормального распределения

© 2000-2025, MetaQuotes Ltd.

Пример:

```
#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;  // логарифм математического ожидания (log mean)
input double std_dev=0.25;    // логарифм среднеквадратического отклонения (log standa
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // количество значений в выборке
   int ncells=51;        // количество интервалов в гистограмме
   double x[];           // центры интервалов гистограммы
   double y[];           // количество значений из выборки, попавших в интервал
   double data[];        // выборка случайных значений
   double max,min;       // максимальное и минимальное значения в выборке
//--- получим выборку из логнормального распределения
   MathRandomLognormal(mean_value,std_dev,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLognormal(x2,mean_value,std_dev,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, 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);
//--- отключим автомасштабирование оси 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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityLognormal

Рассчитывает плотность вероятности логнормального распределения с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // значение случайной величины
   const double  mu,             // логарифм математического ожидания (log mean)
   const double  sigma,          // логарифм среднеквадратического отклонения (log sta
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности логнормального распределения с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // значение случайной величины
   const double  mu,             // логарифм математического ожидания (log mean)
   const double  sigma,          // логарифм среднеквадратического отклонения (log sta
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности логнормального распределения с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает NaN. Аналог dlnorm() в R.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // массив со значениями случайной величины
   const double   mu,            // логарифм математического ожидания (log mean)
   const double   sigma,         // логарифм среднеквадратического отклонения (log st
   const bool     log_mode,      // расчет логарифма значения, если log_mode=true, то
   double&        result[]       // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности логнормального распределения с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // массив со значениями случайной величины
   const double   mu,            // логарифм математического ожидания (log mean)
   const double   sigma,         // логарифм среднеквадратического отклонения (log st
   double&        result[]       // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

mu

[in] Логарифм математического ожидания (log_mean).

sigma

[in] Логарифм среднеквадратического отклонения (log standard deviation).

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для получения значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionLognormal

Рассчитывает логнормальное распределение вероятностей с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // значение случайной величины
   const double  mu,             // логарифм математического ожидания (log mean)
   const double  sigma,          // логарифм среднеквадратического отклонения (log sta
   const bool    tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает логнормальное распределение вероятностей с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // значение случайной величины
   const double  mu,             // логарифм математического ожидания (log mean)
   const double  sigma,          // логарифм среднеквадратического отклонения (log sta
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает логнормальное распределение вероятностей с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false. Аналог plnorm() в R.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // массив со значениями случайной величины
   const double   mu,            // логарифм математического ожидания (log mean)
   const double   sigma,         // логарифм среднеквадратического отклонения (log st
   const bool     tail,          // флаг расчета, если true, то рассчитывается вероят
   const bool     log_mode,      // расчет логарифма значения, если log_mode=true, то
   double&        result[]       // массив для значений функции вероятности
   );
```

Рассчитывает логнормальное распределение вероятностей с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // массив со значениями случайной величины
   const double   mu,            // логарифм математического ожидания (log mean)
   const double   sigma,         // логарифм среднеквадратического отклонения (log st
   double&        result[]       // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

mu

[in] Логарифм математического ожидания (log_mean).

sigma

[in] Логарифм среднеквадратического отклонения (log standard deviation).

tail

[in] Флаг расчета, если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для получения значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileLognormal

Рассчитывает значение обратной функции логнормального распределения с параметрами mu и sigma для вероятности probability. В случае ошибки возвращает NaN.

```
double  MathQuantileLognormal(
   const double  probability,     // значение вероятности появления случайной величины
   const double  mu,              // логарифм математического ожидания (log mean)
   const double  sigma,           // логарифм среднеквадратического отклонения (log sta
   const bool    tail,            // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,        // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code       // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции логнормального распределения с параметрами mu и sigma для вероятности probability. В случае ошибки возвращает NaN.

```
double  MathQuantileLognormal(
   const double  probability,     // значение вероятности появления случайной величины
   const double  mu,              // логарифм математического ожидания (log mean)
   const double  sigma,           // логарифм среднеквадратического отклонения (log sta
   int&          error_code       // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции логнормального распределения с параметрами mu и sigma для массива значений вероятности probability[]. В случае ошибки возвращает false. Аналог qlnorm() в R.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // массив со значениями вероятностей случайной велич
   const double   mu,             // логарифм математического ожидания (log mean)
   const double   sigma,          // логарифм среднеквадратического отклонения (log st
   const bool     tail,           // флаг расчета, если false, то расчет ведется для в
   const bool     log_mode,       // флаг расчета, если log_mode=true, то расчет ведет
   double&        result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции логнормального распределения с параметрами mu и sigma для массива значений вероятности probability[]. В случае ошибки возвращает false.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // массив со значениями вероятностей случайной велич
   const double   mu,             // логарифм математического ожидания (log mean)
   const double   sigma,          // логарифм среднеквадратического отклонения (log st
   double&        result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности появления случайной величины.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Массив со значениями вероятностей случайной величины.

mu

[in] Логарифм математического ожидания (log_mean).

sigma

[in] Логарифм среднеквадратического отклонения (log standard deviation).

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomLognormal

Генерирует псевдослучайную величину, распределенную по логнормальному закону с параметрами mu и sigma. В случае ошибки возвращает NaN.

```
double  MathRandomLognormal(
   const double  mu,              // логарифм математического ожидания (log mean)
   const double  sigma,           // логарифм среднеквадратического отклонения (log sta
   int&          error_code       // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по логнормальному закону с параметрами mu и sigma. В случае ошибки возвращает false. Аналог rlnorm() в R.

```
double  MathRandomLognormal(
   const double  mu,              // логарифм математического ожидания (log mean)
   const double  sigma,           // логарифм среднеквадратического отклонения (log sta
   const int     data_count,      // количество необходимых данных
   double&       result[]         // массив со значениями псевдослучайных величин
   );
```

Параметры

mu

[in] Логарифм математического ожидания (log_mean).

sigma

[in] Логарифм среднеквадратического отклонения (log standard deviation).

data_count

[in] Количество необходимых данных.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив со значениями псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsLognormal

Рассчитывает теоретические численные значения первых 4 моментов логнормального распределения. Возвращает true, если расчет моментов произведен успешно, иначе false.

```
double  MathMomentsLognormal(
   const double  mu,              // логарифм математического ожидания (log mean)
   const double  sigma,           // логарифм среднеквадратического отклонения (log sta
   double&       mean,            // переменная для среднего значения
   double&       variance,        // переменная для дисперсии
   double&       skewness,        // переменная для коэффициента асимметрии
   double&       kurtosis,        // переменная для коэффициента эксцесса
   int&          error_code       // переменная для записи кода ошибки
   );
```

Параметры

mu

[in] Логарифм математического ожидания (log_mean).

sigma

[in] Логарифм среднеквадратического отклонения (log standard deviation).

mean

[in] Переменная для среднего значения.

variance

[out] Переменная для дисперсии.

skewness

[out] Переменная для коэффициента асимметрии.

kurtosis

[out] Переменная для коэффициента эксцесса.

error code

[out] Переменная для записи кода ошибки.

Возвращаемое значение

Возвращает true, если моменты успешно рассчитаны, иначе false.

© 2000-2025, MetaQuotes Ltd.

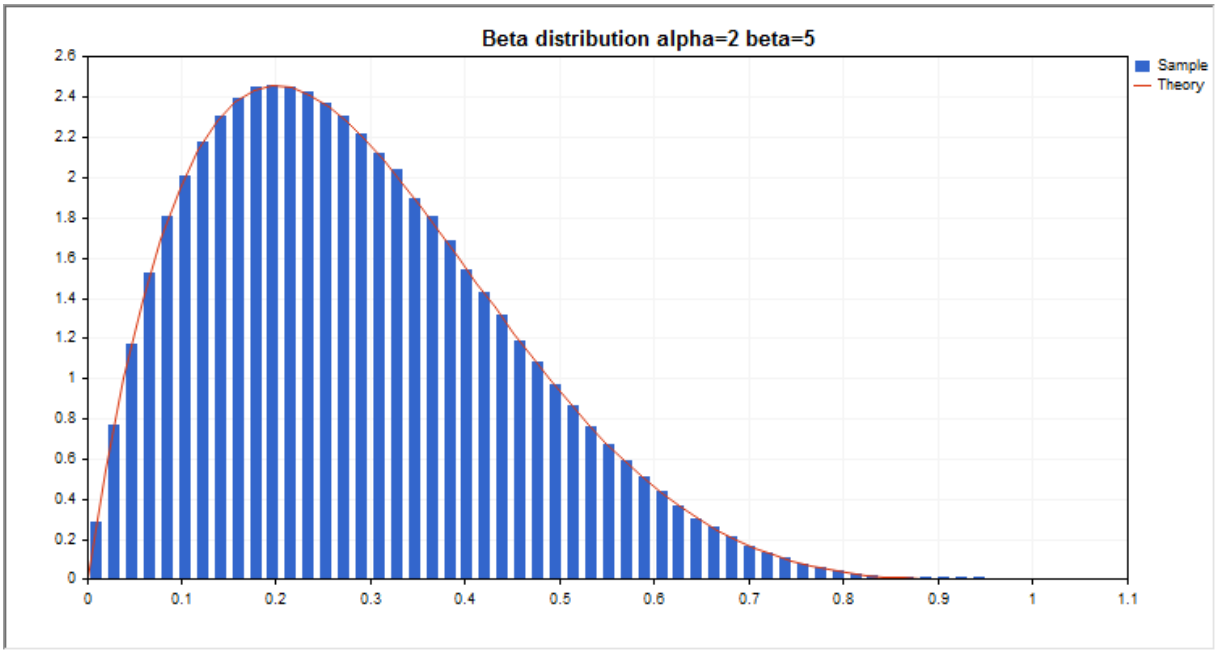
Бета-распределение

В данном разделе представлены функции для работы с бета-распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по соответствующему закону. Бета-распределение описывается следующей формулой:

$$f_{Beta}(x \mid a,b) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

где:

- x — значение случайной величины
- a — первый параметр бета-распределения
- b — второй параметр бета-распределения



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityBeta	Рассчитывает плотность вероятности бета-распределения
MathCumulativeDistributionBeta	Рассчитывает значение функции бета-распределения вероятностей
MathQuantileBeta	Рассчитывает значение обратной функции бета-распределения для заданной вероятности
MathRandomBeta	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону бета-распределения

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathMomentsBeta	Рассчитывает теоретические численные значения первых 4 моментов бета-распределения

Пример:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Beta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double alpha=2;   // первый параметр бета-распределения (shape1)
input double beta=5;    // второй параметр бета-распределения (shape2)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // количество значений в выборке
   int ncells=51;       // количество интервалов в гистограмме
   double x[];          // центры интервалов гистограммы
   double y[];          // количество значений из выборки, попавших в интервал
   double data[];       // выборка случайных значений
   double max,min;      // максимальное и минимальное значения в выборке
//--- получим выборку из бета-распределения
   MathRandomBeta(alpha,beta,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityBeta(x2,alpha,beta,false,y2);
//--- масштабируем
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityBeta

Рассчитывает плотность вероятности бета-распределения с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности бета-распределения с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности бета-распределения с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dbeta() в R.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значения функции плотности вероятности
   );
```

Рассчитывает плотность вероятности бета-распределения с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   double&       result[]        // массив для значения функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

a

[in] Первый параметр бета-распределения (shape 1).

b

[in] Второй параметр бета-распределения (shape 2)

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значения функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionBeta

Рассчитывает распределение вероятностей бета-распределения с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const bool    tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей бета-распределения с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей бета-распределения с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pbeta() в R.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const bool    tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значения функции вероятности
   );
```

Рассчитывает распределение вероятностей бета-распределения с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

a

[in] Первый параметр бета-распределения (shape 1).

b

[in] Второй параметр бета-распределения (shape 2)

tail

[in] Флаг расчета, если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileBeta

Рассчитывает для вероятности *probability* значение обратной функции бета-распределения с параметрами a и b. В случае ошибки возвращает NaN.

```
double  MathQuantileBeta(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает для вероятности *probability* значение обратной функции бета-распределения с параметрами a и b. В случае ошибки возвращает NaN.

```
double  MathQuantileBeta(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает для массива значений вероятности *probability[]* значения обратной функции бета-распределения с параметрами a и b. В случае ошибки возвращает false. Аналог qbeta() в R.

```
double  MathQuantileBeta(
   const double&  probability[], // массив со значениями вероятностей случайной величи
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает для массива значений вероятности *probability[]* значения обратной функции бета-распределения с параметрами a и b. В случае ошибки возвращает false.

```
bool  MathQuantileBeta(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями вероятностей случайной величины.

a

[in] Первый параметр бета-распределения (shape1).

b

[in] Второй параметр бета-распределения (shape2).

tail

[in] Флаг расчета, если lower_tail=false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomBeta

Генерирует псевдослучайную величину, распределенную по закону бета-распределения с параметрами a и b. В случае ошибки возвращает NaN.

```
double  MathRandomBeta(
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону бета-распределения с параметрами a и b. В случае ошибки возвращает false. Аналог rbeta() в R.

```
bool  MathRandomBeta(
   const double  a,               // первый параметр бета-распределения (shape1)
   const double  b,               // второй параметр бета-распределения (shape2)
   const int     data_count,      // количество необходимых данных
   double&       result[]         // массив для получения псевдослучайных величин
   );
```

Параметры

a

[in] Первый параметр бета-распределения (shape1)

b

[in] Второй параметр бета-распределения (shape2).

data_count

[in] Количество псевдослучайных значений, которые необходимо получить.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsBeta

Рассчитывает теоретические численные значения первых 4 моментов бета-распределения.

```
double  MathMomentsBeta(
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

a

[in] Первый параметр бета-распределения (shape1).

b

[in] Второй параметр бета-распределения (shape2).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если моменты успешно рассчитаны, иначе false.

© 2000-2025, MetaQuotes Ltd.

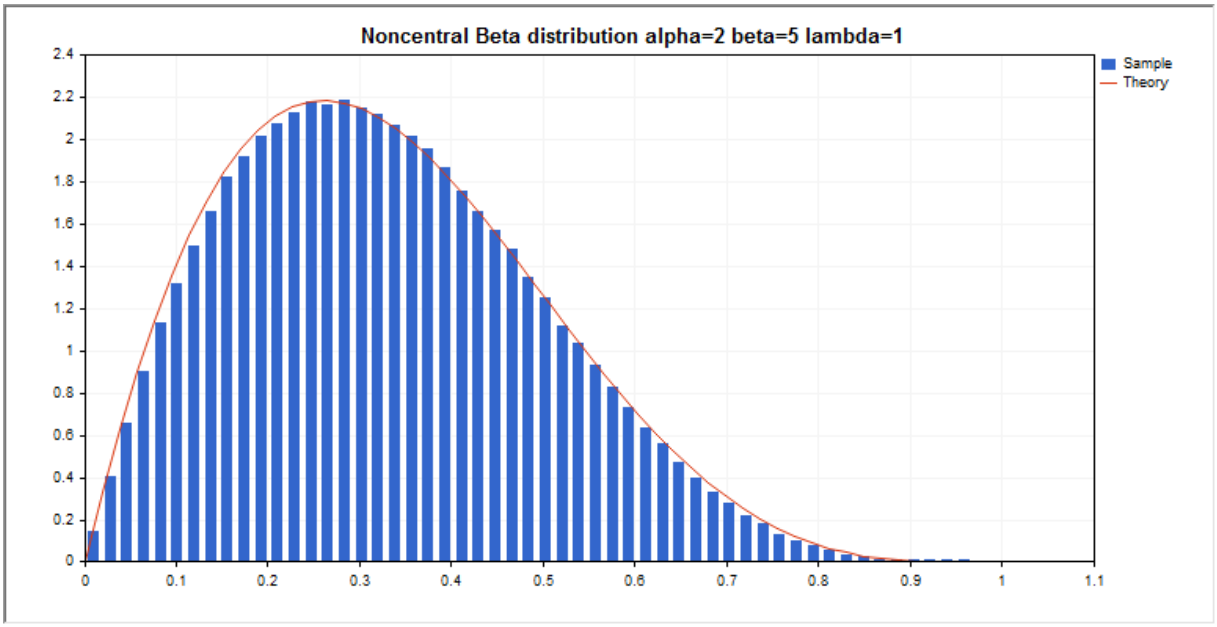
Нецентральное бета-распределение

В данном разделе представлены функции для работы с нецентральным бета-распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по соответствующему закону. Нецентральное бета-распределение описывается следующей формулой:

$$f_{NoncentralBeta}(x|a,b,\lambda)=\sum_{r=0}^{\infty}e^{-\frac{\lambda}{2}}\frac{\left(\frac{\lambda}{2}\right)^{r}}{r!}\frac{x^{a+r-1}(1-x)^{b-1}}{B(a+r,b)}$$

где:

- x — значение случайной величины
- a — первый параметр бета-распределения
- b — второй параметр бета-распределения
- λ — параметр нецентральности



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityNoncentralBeta	Рассчитывает плотность вероятности нецентрального бета-распределения
MathCumulativeDistributionNoncentralBeta	Рассчитывает значение функции нецентрального бета-распределения вероятностей
MathQuantileNoncentralBeta	Рассчитывает значение обратной функции нецентрального бета-распределения для заданной вероятности
MathRandomNoncentralBeta	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону нецентрального бета-

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
	распределения
MathMomentsNoncentralBeta	Рассчитывает теоретические численные значения первых 4 моментов нецентрального бета-распределения

Пример:

```
#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;     // первый параметр бета-распределения (shape1)
input double b_par=5;     // второй параметр бета-распределения (shape2)
input double l_par=1;     // параметр нецентральности (lambda)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // количество значений в выборке
   int ncells=53;          // количество интервалов в гистограмме
   double x[];             // центры интервалов гистограммы
   double y[];             // количество значений из выборки, попавших в интервал
   double data[];          // выборка случайных значений
   double max,min;         // максимальное и минимальное значения в выборке
//--- получим выборку из нецентрального бета-распределения
   MathRandomNoncentralBeta(a_par,b_par,l_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralBeta(x2,a_par,b_par,l_par,false,y2);
//--- масштабируем
```

© 2000-2025, MetaQuotes Ltd.

```
   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;
//--- выводим графики
   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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

© 2000-2025, MetaQuotes Ltd.

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralBeta

Рассчитывает плотность вероятности нецентрального бета-распределения с параметрами a, b и lambda для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности нецентрального бета-распределения с параметрами a, b и lambda для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности нецентрального бета-распределения с параметрами a, b и lambda для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dbeta() в R.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности нецентрального бета-распределения с параметрами a, b и lambda для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

© 2000-2025, MetaQuotes Ltd.

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

a

[in] Первый параметр бета-распределения (shape 1).

b

[in] Второй параметр бета-распределения (shape 2)

lambda

[in] Параметр нецентральности

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralBeta

Рассчитывает распределение вероятностей нецентрального бета-распределения с параметрами a, b и lambda для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   const bool    tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей нецентрального бета-распределения с параметрами a, b и lambda для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей нецентрального бета-распределения с параметрами a, b и lambda для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pbeta() в R.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   const bool    tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает распределение вероятностей нецентрального бета-распределения с параметрами a, b и lambda для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   double&       result[]        // массив для значений функции вероятности
   );
```

© 2000-2025, MetaQuotes Ltd.

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

a

[in] Первый параметр бета-распределения (shape 1).

b

[in] Второй параметр бета-распределения (shape 2)

lambda

[in] Параметр нецентральности

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralBeta

Рассчитывает значение обратной функции распределения вероятностей нецентрального бета-распределения с параметрами a, b и lambda для вероятности появления значения случайной величины *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции распределения вероятностей нецентрального бета-распределения с параметрами a, b и lambda для вероятности появления значения случайной величины *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции распределения вероятностей нецентрального бета-распределения с параметрами a, b и lambda для массива значений вероятности probability[]. В случае ошибки возвращает false. Аналог qbeta() в R.

```
double  MathQuantileNoncentralBeta(
   const double&  probability[],  // массив со значениями вероятностей случайной велич
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции распределения вероятностей нецентрального бета-распределения с параметрами a, b и lambda для массива значений вероятности probability[]. В случае ошибки возвращает false.

```
bool  MathQuantileNoncentralBeta(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
```

© 2000-2025, MetaQuotes Ltd.

```
   const double  lambda,          // параметр нецентральности
   double&       result[]         // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

a

[in] Первый параметр бета-распределения (shape1).

b

[in] Второй параметр бета-распределения (shape2).

lambda

[in] Параметр нецентральности.

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralBeta

Генерирует псевдослучайную величину, распределенную по закону нецентрального бета-распределения с параметрами a, b и lambda. В случае ошибки возвращает NaN.

```
double  MathRandomNoncentralBeta(
   const double  a,             // первый параметр бета-распределения (shape1)
   const double  b,             // второй параметр бета-распределения (shape2)
   const double  lambda,        // параметр нецентральности
   int&          error_code     // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону нецентрального бета-распределения с параметрами a, b и lambda. В случае ошибки возвращает false. Аналог rbeta() в R.

```
bool  MathRandomNoncentralBeta(
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив для получения псевдослучайных величин
   );
```

Параметры

a

[in] Первый параметр бета-распределения (shape1)

b

[in] Второй параметр бета-распределения (shape2).

lambda

[in] Параметр нецентральности

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralBeta

Рассчитывает теоретические численные значения первых 4 моментов нецентрального бета-распределения с параметрами a, b и lambda.

```
double  MathMomentsNoncentralBeta(
   const double  a,              // первый параметр бета-распределения (shape1)
   const double  b,              // второй параметр бета-распределения (shape2)
   const double  lambda,         // параметр нецентральности
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

a

[in] Первый параметр бета-распределения (shape1).

b

[in] Второй параметр бета-распределения (shape2).

lambda

[in] Параметр нецентральности

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

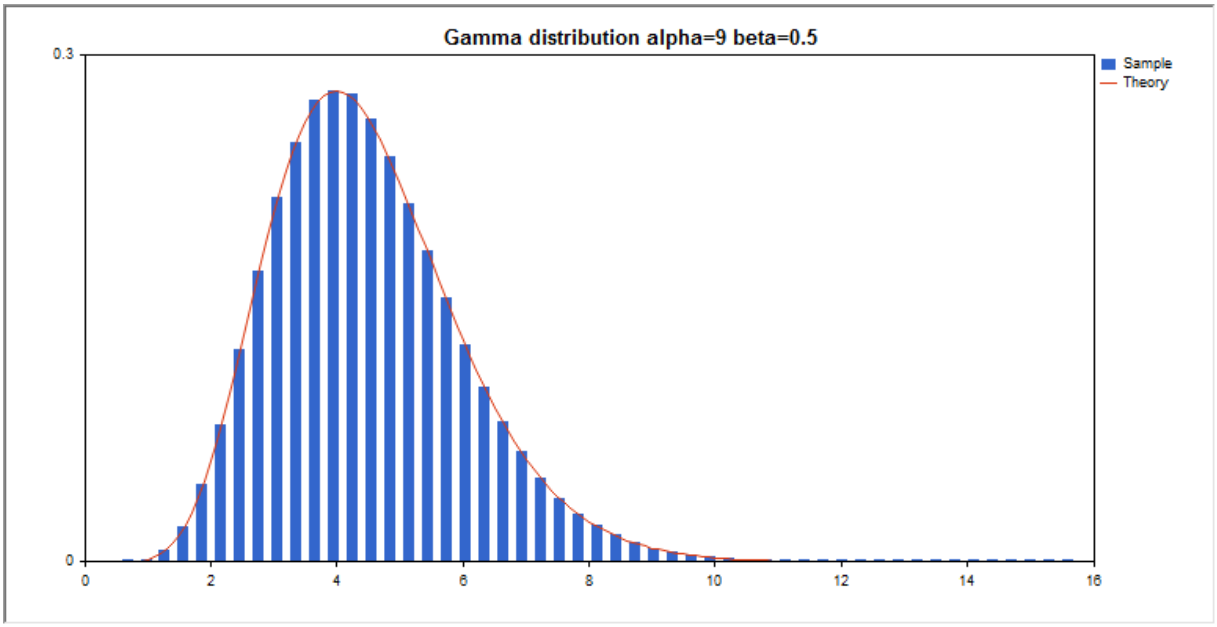
Гамма-распределение

В данном разделе представлены функции для работы с гамма-распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по соответствующему закону. Гамма-распределение описывается следующей формулой:

$$f_{Gamma}(x|a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-\frac{x}{b}}$$

где:

- x — значение случайной величины
- a — первый параметр распределения
- b — второй параметр распределения



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityGamma	Рассчитывает плотность вероятности гамма-распределения
MathCumulativeDistributionGamma	Рассчитывает значение функции гамма-распределения вероятностей
MathQuantileGamma	Рассчитывает значение обратной функции гамма-распределения для заданной вероятности
MathRandomGamma	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону гамма-распределения
MathMomentsGamma	Рассчитывает теоретические численные значения первых 4 моментов гамма-распределения

© 2000-2025, MetaQuotes Ltd.

Пример:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Gamma.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double alpha=9;   // первый параметр гамма-распределения (shape)
input double beta=0.5;  // второй параметр гамма-распределения (scale)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // количество значений в выборке
   int ncells=51;        // количество интервалов в гистограмме
   double x[];           // центры интервалов гистограммы
   double y[];           // количество значений из выборки, попавших в интервал
   double data[];        // выборка случайных значений
   double max,min;       // максимальное и минимальное значения в выборке
//--- получим выборку из гамма-распределения
   MathRandomGamma(alpha,beta,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityGamma(x2,alpha,beta,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Gamma distribution alpha=%G beta=%G",alpha,bet
   graphic.BackgroundMainSize(16);
//--- отключим автомасштабирование оси 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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityGamma

Рассчитывает плотность вероятности гамма-распределения с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности гамма-распределения с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности гамма-распределения с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dgamma() в R.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности гамма-распределения с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

a

[in] Первый параметр распределения (shape).

b

[in] Второй параметр распределения (scale).

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionGamma

Рассчитывает гамма-распределение вероятностей с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает гамма-распределение вероятностей с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // значение случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает гамма-распределение вероятностей с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pgamma() в R.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает гамма-распределение вероятностей с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

a

[in] Первый параметр распределения (shape).

b

[in] Второй параметр распределения (scale)

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileGamma

Рассчитывает значение обратной функции гамма-распределения с параметрами a и b для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileGamma(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции гамма-распределения с параметрами a и b для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileGamma(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции гамма-распределения с параметрами a и b для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qgamma() в R.

```
double  MathQuantileGamma(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции гамма-распределения с параметрами a и b для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileGamma(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями вероятностей случайной величины.

a

[in] Первый параметр распределения (shape).

b

[in] Второй параметр распределения (scale).

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomGamma

Генерирует псевдослучайную величину, распределенную по закону гамма-распределения с параметрами a и b. В случае ошибки возвращает NaN.

```
double  MathRandomGamma(
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону гамма-распределения с параметрами a и b. В случае ошибки возвращает false. Аналог rgamma() в R.

```
bool  MathRandomGamma(
   const double  a,               // первый параметр распределения (shape)
   const double  b,               // второй параметр распределения (scale)
   const int     data_count,      // количество необходимых данных
   double&       result[]         // массив со значениями псевдослучайных величин
   );
```

Параметры

a

[in] Первый параметр распределения (shape).

b

[in] Второй параметр распределения (scale).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsGamma

Рассчитывает теоретические численные значения первых 4 моментов гамма-распределения с параметрами a, b.

```
double  MathMomentsGamma(
   const double  a,              // первый параметр распределения (shape)
   const double  b,              // второй параметр распределения (scale)
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

a

[in] Первый параметр распределения (shape).

b

[in] Второй параметр распределения (scale).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

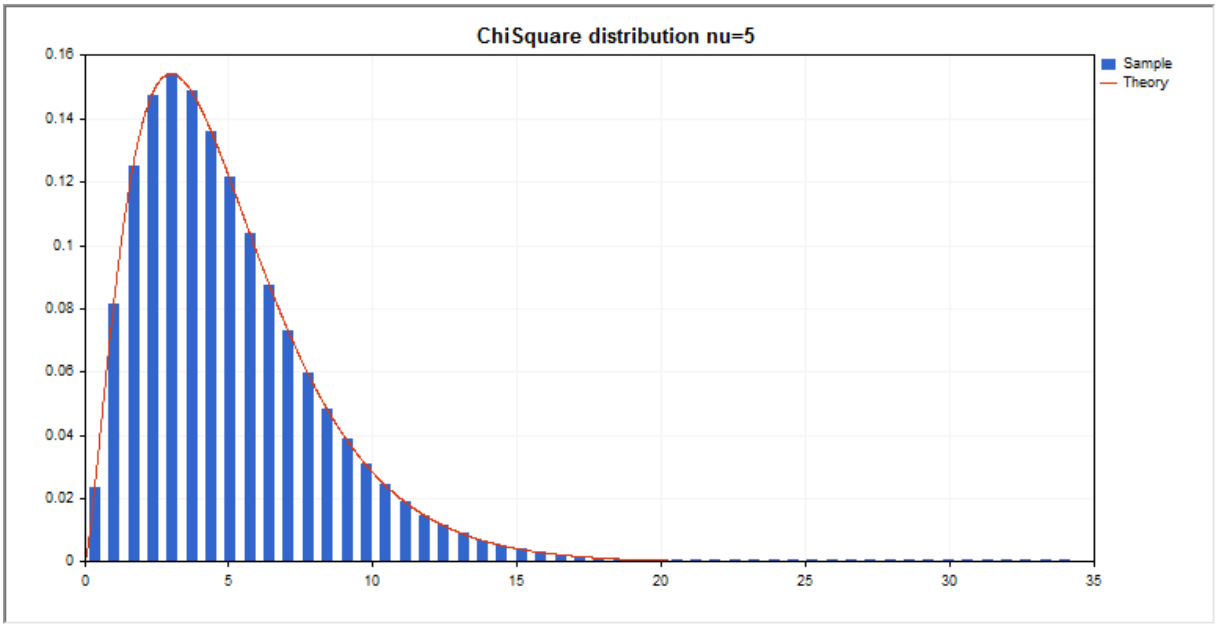
Распределение хи-квадрат

В данном разделе представлены функции для работы с распределением хи-квадрат. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по соответствующему закону. Распределение хи-квадрат описывается следующей формулой:

$$f_{Chi-Square}(x|\nu) = \frac{x^{\frac{(\nu-2)}{2}} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)}$$

где:

- x — значение случайной величины
- ν — число степеней свободы



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityChiSquare	Рассчитывает плотность вероятности распределения хи-квадрат
MathCumulativeDistributionChiSquare	Рассчитывает значение функции распределения вероятностей хи-квадрат
MathQuantileChiSquare	Рассчитывает значение обратной функции распределения хи-квадрат для заданной вероятности

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathRandomChiSquare	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону распределения хи-квадрат
MathMomentsChiSquare	Рассчитывает теоретические численные значения первых 4 моментов распределения хи-квадрат

Пример:

```
#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;     // число степеней свободы
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // количество значений в выборке
   int ncells=51;        // количество интервалов в гистограмме
   double x[];           // центры интервалов гистограммы
   double y[];           // количество значений из выборки, попавших в интервал
   double data[];        // выборка случайных значений
   double max,min;       // максимальное и минимальное значения в выборке
//--- получим выборку из хи-квадрат распределения
   MathRandomChiSquare(nu_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityChiSquare(x2,nu_par,false,y2);
//--- масштабируем
   double theor_max=y2[ArrayMaximum(y2)];
```

© 2000-2025, MetaQuotes Ltd.

```
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityChiSquare

Рассчитывает плотность вероятности распределения хи-квадрат с параметром nu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности распределения хи-квадрат с параметром nu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности распределения хи-квадрат с параметром nu для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dchisq() в R.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности распределения хи-квадрат с параметром nu для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

nu

[in] Параметр распределения (число степеней свободы)

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionChiSquare

Рассчитывает распределение вероятностей хи-квадрат с параметром nu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей хи-квадрат с параметром nu для случайной величины x. В случае ошибки возвращает NaN

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей хи-квадрат с параметром nu для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pchisq() в R.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает распределение вероятностей хи-квадрат с параметром nu для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileChiSquare

Рассчитывает значение обратной функции распределения вероятностей хи-квадрат для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции распределения вероятностей хи-квадрат для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции распределения вероятностей хи-квадрат для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qchisq() в R.

```
double  MathQuantileChiSquare(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  nu,             // параметр распределения (число степеней свободы)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции распределения вероятностей хи-квадрат для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileChiSquare(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  nu,             // параметр распределения (число степеней свободы)
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomChiSquare

Генерирует псевдослучайную величину, распределенную по закону распределения хи-квадрат с параметром nu. В случае ошибки возвращает NaN.

```
double  MathRandomChiSquare(
   const double  nu,            // параметр распределения (число степеней свободы)
   int&          error_code     // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону распределения хи-квадрат с параметром nu. В случае ошибки возвращает false. Аналог rchisq() в R.

```
bool  MathRandomChiSquare(
   const double  nu,             // параметр распределения (число степеней свободы)
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

nu

[in] Параметр распределения (число степеней свободы).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsChiSquare

Рассчитывает теоретические численные значения первых 4 моментов распределения хи-квадрат с параметром nu.

```
double  MathMomentsChiSquare(
   const double  nu,              // параметр распределения (число степеней свободы)
   double&       mean,            // переменная для среднего значения
   double&       variance,        // переменная для дисперсии
   double&       skewness,        // переменная для коэффициента асимметрии
   double&       kurtosis,        // переменная для коэффициента эксцесса
   int&          error_code       // переменная для кода ошибки
   );
```

Параметры

nu

[in] Параметр распределения (число степеней свободы).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

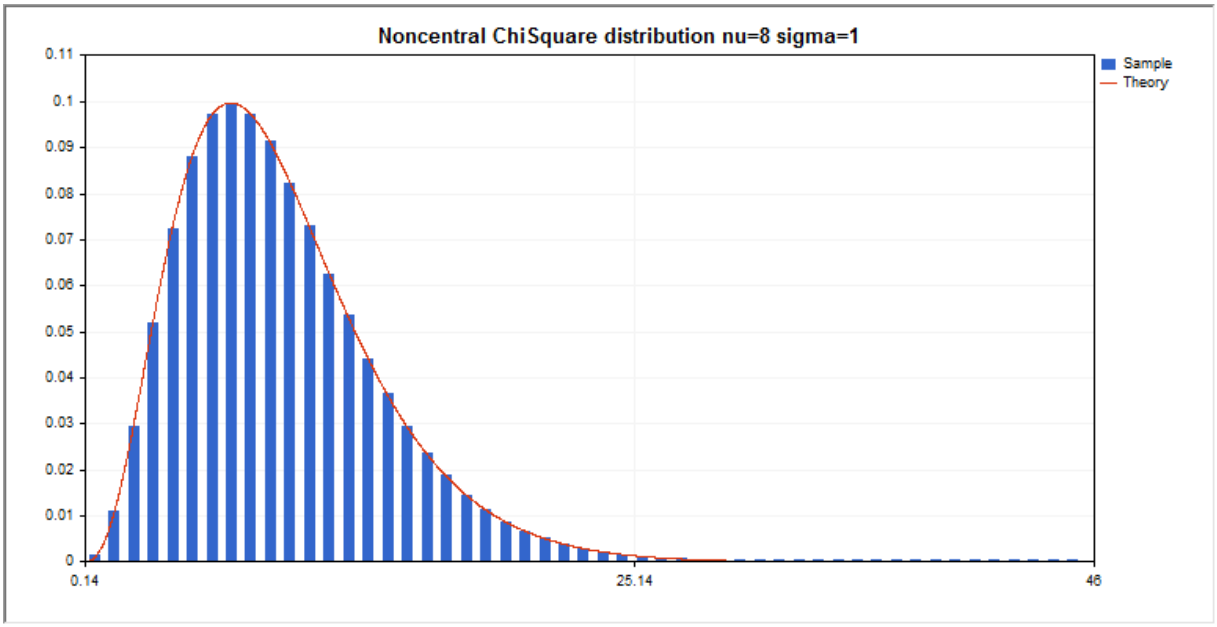
Нецентральное распределение хи-квадрат

В данном разделе представлены функции для работы с нецентральным распределением хи-квадрат. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по соответствующему закону. Нецентральное распределение хи-квадрат описывается следующей формулой:

$$f_{NoncentralChiSquare}(x|\nu,\sigma)=\frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{1}{2}\right)}x^{\frac{\nu}{2}-1}e^{-\frac{(x+\sigma)}{2}}\sum_{r=0}^{\infty}\frac{(\lambda x)^r}{(2r)!}\frac{\Gamma\left(\frac{1}{2}+r\right)}{\Gamma\left(\frac{\nu}{2}+r\right)}$$

где:

- x — значение случайной величины
- ν — число степеней свободы
- σ — параметр нецентральности



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityNoncentralChiSquare	Рассчитывает плотность вероятности нецентрального распределения хи-квадрат
MathCumulativeDistributionNoncentralChiSquare	Рассчитывает значение функции нецентрального распределения вероятностей хи-квадрат
MathQuantileNoncentralChiSquare	Рассчитывает значение обратной функции нецентрального распределения хи-квадрат для заданной вероятности

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathRandomNoncentralChiSquare	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону нецентрального распределения хи-квадрат
MathMomentsNoncentralChiSquare	Рассчитывает теоретические численные значения первых 4 моментов нецентрального распределения хи-квадрат

Пример:

```
#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;     // число степеней свободы
input double si_par=1;     // параметр нецентральности
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // количество значений в выборке
   int ncells=51;          // количество интервалов в гистограмме
   double x[];             // центры интервалов гистограммы
   double y[];             // количество значений из выборки, попавших в интервал
   double data[];          // выборка случайных значений
   double max,min;         // максимальное и минимальное значения в выборке
//--- получим выборку из нецентрального хи-квадрат распределения
   MathRandomNoncentralChiSquare(nu_par,si_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralChiSquare(x2,nu_par,si_par,false,y2);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- масштабируем
   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;
//--- выводим графики
   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);
//--- отключим автомасштабирование оси 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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
```

© 2000-2025, MetaQuotes Ltd.

```
   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)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralChiSquare

Рассчитывает плотность вероятности нецентрального распределения хи-квадрат с параметрами nu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности нецентрального распределения хи-квадрат с параметрами nu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности нецентрального распределения хи-квадрат с параметрами nu и sigma для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dchisq() в R.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности нецентрального распределения хи-квадрат с параметром nu для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralChiSquare

Рассчитывает распределение вероятностей нецентрального распределения хи-квадрат с параметрами nu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   const bool    tail,           // флаг расчета, если lower_tail=true, то рассчитывает
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей нецентрального распределения хи-квадрат с параметрами nu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей нецентрального распределения хи-квадрат с параметрами nu и sigma для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pchisq() в R.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   const bool    tail,           // флаг расчета, если lower_tail=true, то рассчитывае
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает распределение вероятностей нецентрального распределения хи-квадрат с параметрами nu и sigma для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

© 2000-2025, MetaQuotes Ltd.

x[]

[in] Массив со значениями случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralChiSquare

Рассчитывает значение обратной функции нецентрального распределения хи-квадрат с параметрами nu и sigma для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции нецентрального распределения хи-квадрат с параметрами nu и sigma для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции нецентрального распределения хи-квадрат с параметрами nu и sigma для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qchisq() в R.

```
double  MathQuantileNoncentralChiSquare(
   const double&  probability[],  // массив со значениями вероятностей случайной велич
   const double  nu,              // параметр распределения (число степеней свободы)
   const double  sigma,           // параметр нецентральности
   const bool    tail,            // флаг расчета, если false, то расчет ведется для в
   const bool    log_mode,        // флаг расчета, если log_mode=true, то расчет ведет
   double&       result[]         // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции нецентрального распределения вероятностей хи-квадрат для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileNoncentralChiSquare(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Массив со значениями вероятностей случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralChiSquare

Генерирует псевдослучайную величину, распределенную по закону нецентрального распределения хи-квадрат с параметрами nu и sigma. В случае ошибки возвращает NaN.

```
double  MathRandomNoncentralChiSquare(
   const double  nu,            // параметр распределения (число степеней свободы)
   const double  sigma,         // параметр нецентральности
   int&          error_code     // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону нецентрального распределения хи-квадрат с параметрами nu и sigma. В случае ошибки возвращает false. Аналог rchisq() в R.

```
bool  MathRandomNoncentralChiSquare(
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  sigma,          // параметр нецентральности
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

nu

[in] Параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralChiSquare

Рассчитывает теоретические численные значения первых 4 моментов нецентрального распределения хи-квадрат с параметрами nu и sigma.

```
double  MathMomentsNoncentralChiSquare(
   const double  nu,              // параметр распределения (число степеней свободы)
   const double  sigma,           // параметр нецентральности
   double&       mean,            // переменная для среднего значения
   double&       variance,        // переменная для дисперсии
   double&       skewness,        // переменная для коэффициента асимметрии
   double&       kurtosis,        // переменная для коэффициента эксцесса
   int&          error_code       // переменная для кода ошибки
   );
```

Параметры

nu

[in] Параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

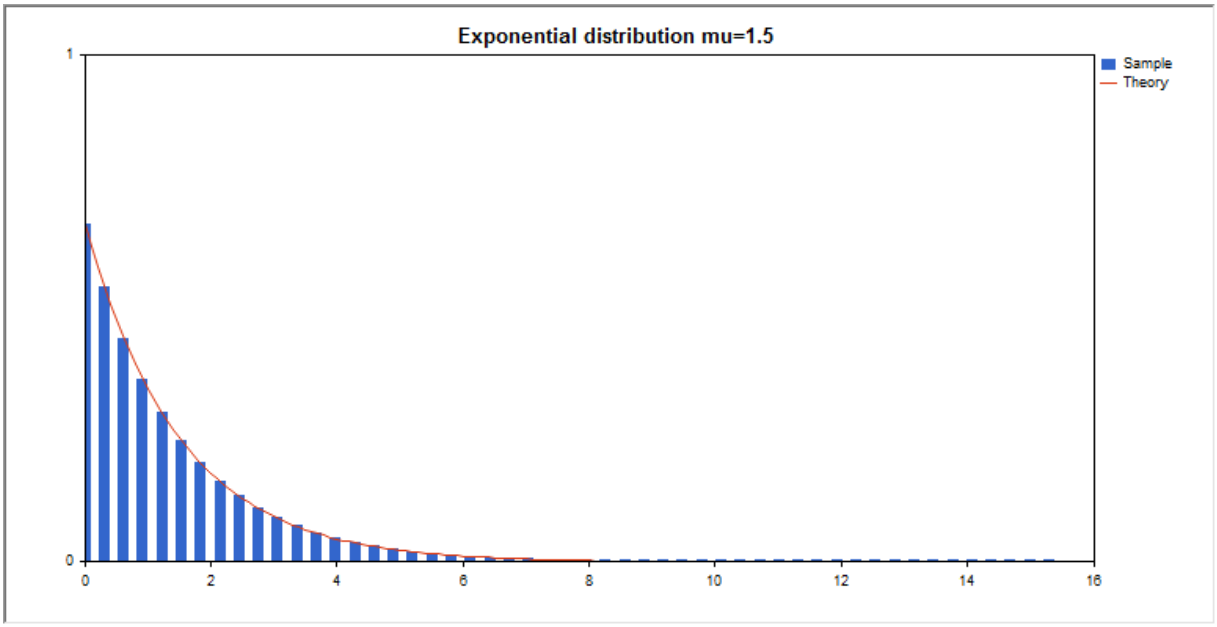
Экспоненциальное

В данном разделе представлены функции для работы с экспоненциальным распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по экспоненциальному закону. Экспоненциальное распределение описывается следующей формулой:

$$f_{Exponential}(x \mid \mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

где:

- x — значение случайной величины
- μ — математическое ожидание



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityExponential	Рассчитывает плотность вероятности экспоненциального распределения
MathCumulativeDistributionExponential	Рассчитывает значение функции экспоненциального распределения вероятностей
MathQuantileExponential	Рассчитывает значение обратной функции экспоненциального распределения для заданной вероятности
MathRandomExponential	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по экспоненциальному закону
MathMomentsExponential	Рассчитывает теоретические численные значения первых 4 моментов экспоненциального распределения

© 2000-2025, MetaQuotes Ltd.

Пример:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Exponential.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double mu_par=1.5;     // число степеней свободы
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // количество значений в выборке
   int ncells=51;          // количество интервалов в гистограмме
   double x[];             // центры интервалов гистограммы
   double y[];             // количество значений из выборки, попавших в интервал
   double data[];          // выборка случайных значений
   double max,min;         // максимальное и минимальное значения в выборке
//--- получим выборку из экспоненциального распределения
   MathRandomExponential(mu_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityExponential(x2,mu_par,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Exponential distribution mu=%G ",mu_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   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)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
```

© 2000-2025, MetaQuotes Ltd.

```
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityExponential

Рассчитывает плотность вероятности экспоненциального распределения с параметром mu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности экспоненциального распределения с параметром mu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности экспоненциального распределения с параметром mu для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dexp() в R.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности экспоненциального распределения с параметром mu для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

mu

[in] Параметр распределения (математическое ожидание)

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionExponential

Рассчитывает экспоненциальное распределение вероятностей с параметром mu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает экспоненциальное распределение вероятностей с параметром mu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает экспоненциальное распределение вероятностей с параметром mu для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pexp() в R.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает экспоненциальное распределение вероятностей с параметром mu для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

mu

[in] Параметр распределения (математическое ожидание).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileExponential

Рассчитывает значение обратной функции экспоненциального распределения вероятностей с параметром mu для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileExponential(
   const double  probability,    // значение вероятности появления случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции экспоненциального распределения вероятностей с параметром mu для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileExponential(
   const double  probability,    // значение вероятности появления случайной величины
   const double  mu,             // параметр распределения (математическое ожидание)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции экспоненциального распределения вероятностей с параметром mu для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qexp() в R.

```
double  MathQuantileExponential(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  mu,             // параметр распределения (математическое ожидание)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции экспоненциального распределения вероятностей с параметром mu для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileExponential(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  mu,             // параметр распределения (математическое ожидание)
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

© 2000-2025, MetaQuotes Ltd.

mu

[in] Параметр распределения (математическое ожидание).

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomExponential

Генерирует псевдослучайную величину, распределенную по закону экспоненциального распределения с параметром mu. В случае ошибки возвращает NaN.

```
double  MathRandomExponential(
   const double  mu,            // параметр распределения (математическое ожидание)
   int&          error_code     // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону экспоненциального распределения с параметром mu. В случае ошибки возвращает false. Аналог rexp() в R.

```
bool  MathRandomExponential(
   const double  mu,             // параметр распределения (математическое ожидание)
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

mu

[in] Параметр распределения (математическое ожидание).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsExponential

Рассчитывает теоретические численные значения первых 4 моментов экспоненциального распределения с параметром mu.

```
double  MathMomentsExponential(
   const double  mu,              // параметр распределения (математическое ожидание)
   double&       mean,            // переменная для среднего значения
   double&       variance,        // переменная для дисперсии
   double&       skewness,        // переменная для коэффициента асимметрии
   double&       kurtosis,        // переменная для коэффициента эксцесса
   int&          error_code       // переменная для кода ошибки
   );
```

Параметры

mu

[in] Параметр распределения (математическое ожидание).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

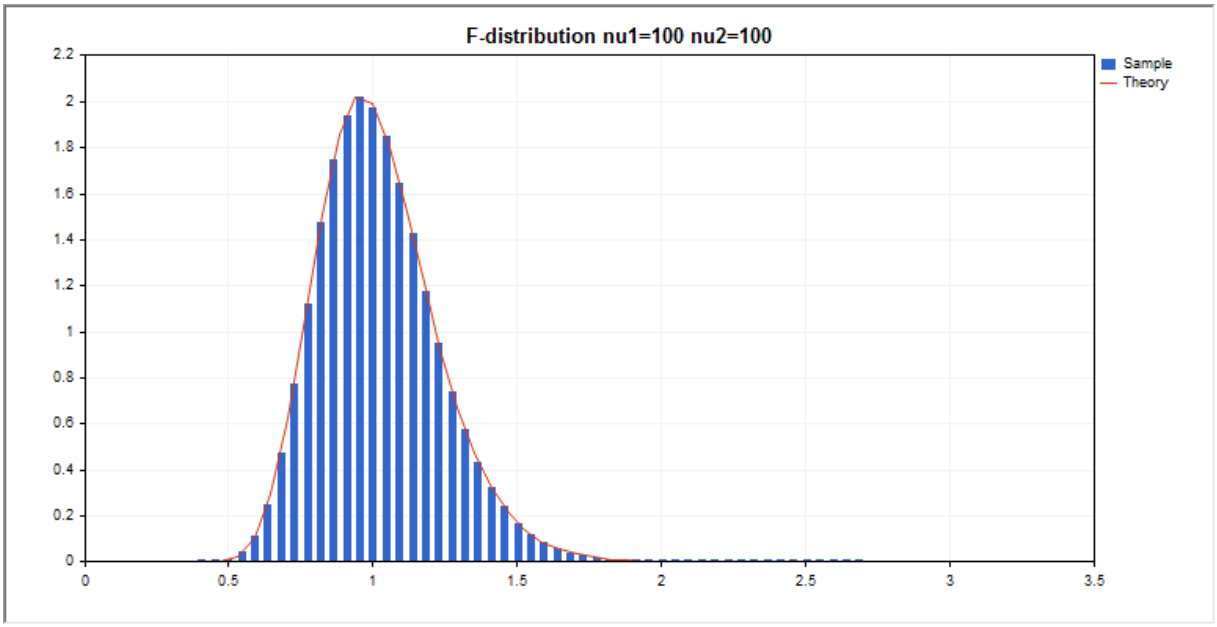
F-распределение

В данном разделе представлены функции для работы с F-распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по закону Фишера. F-распределение описывается следующей формулой:

$$f_F(x|\nu_1,\nu_2)=\frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}\right)}{\Gamma\left(\frac{\nu_1}{2}\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}}\frac{x^{\frac{\nu_1-2}{2}}}{\left(1+\left(\frac{\nu_1}{\nu_2}\right)x\right)^{\frac{\nu_1+\nu_2}{2}}}$$

где:

- x — значение случайной величины
- ν1 — первый параметр распределения (число степеней свободы)
- ν2 — второй параметр распределения (число степеней свободы)



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityF	Рассчитывает плотность вероятности F-распределения
MathCumulativeDistributionF	Рассчитывает значение функции F-распределения вероятностей
MathQuantileF	Рассчитывает значение обратной функции F-распределения для заданной вероятности
MathRandomF	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону Фишера

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathMomentsF	Рассчитывает теоретические численные значения первых 4 моментов распределения Фишера

Пример:

```
#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;     // первое число степеней свободы
input double nu_2=100;     // второе число степеней свободы
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // количество значений в выборке
   int ncells=51;        // количество интервалов в гистограмме
   double x[];           // центры интервалов гистограммы
   double y[];           // количество значений из выборки, попавших в интервал
   double data[];        // выборка случайных значений
   double max,min;       // максимальное и минимальное значения в выборке
//--- получим выборку из F-распределения
   MathRandomF(nu_1,nu_2,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityF(x2,nu_1,nu_2,false,y2);
//--- масштабируем
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- выводим графики
   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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityF

Рассчитывает плотность вероятности F-распределения Фишера с параметрами nu1 и nu2 для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // значение случайной величины
   const double  nu1,            // первый параметр распределения (число степеней свобо
   const double  nu2,            // второй параметр распределения (число степеней свобо
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности F-распределения Фишера с параметрами nu1 и nu2 для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // значение случайной величины
   const double  nu1,            // первый параметр распределения (число степеней свобо
   const double  nu2,            // второй параметр распределения (число степеней свобо
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности F-распределения Фишера с параметрами nu1 и nu2 для массива случайных величин x[]. В случае ошибки возвращает false. Аналог df() в R.

```
bool  MathProbabilityDensityF(
   const double& x[],            // массив со значениями случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности F-распределения Фишера с параметрами nu1 и nu2 для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityF(
   const double& x[],            // массив со значениями случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionF

Рассчитывает распределение вероятностей по закону F-распределения Фишера с параметрами nu1 и nu2 для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionF(
   const double  x,              // значение случайной величины
   const double  nu1,            // первый параметр распределения (число степеней свобо
   const double  nu2,            // второй параметр распределения (число степеней свобо
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей по закону F-распределения Фишера с параметрами nu1 и nu2 для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionF(
   const double  x,              // значение случайной величины
   const double  nu1,            // первый параметр распределения (число степеней свобо
   const double  nu2,            // второй параметр распределения (число степеней свобо
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей по закону F-распределения Фишера с параметрами nu1 и nu2 для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pf() в R.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // массив со значениями случайной величины
   const double  nu1,            // первый параметр распределения (число степеней сво
   const double  nu2,            // второй параметр распределения (число степеней сво
   const double  tail,           // флаг расчета, если true, то рассчитывается вероят
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает распределение вероятностей по закону F-распределения Фишера с параметрами nu1 и nu2 для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // массив со значениями случайной величины
   const double  nu1,            // первый параметр распределения (число степеней сво
   const double  nu2,            // второй параметр распределения (число степеней сво
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileF

Рассчитывает значение обратной функции F-распределения вероятностей с параметрами nu1 и nu2 для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileF(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции F-распределения вероятностей с параметрами nu1 и nu2 для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileF(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции F-распределения вероятностей с параметрами nu1 и nu2 для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qf() в R.

```
double  MathQuantileF(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции F-распределения вероятностей с параметрами nu1 и nu2 для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileF(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Массив со значениями вероятностей случайной величины.

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

tail

[in] Флаг расчета, если lower_tail=false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomF

Генерирует псевдослучайную величину, распределенную по закону F-распределения Фишера с параметрами nu1 и nu2. В случае ошибки возвращает NaN.

```
double  MathRandomF(
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   int&          error_code     // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, по закону F-распределения Фишера с параметрами nu1 и nu2. В случае ошибки возвращает false. Аналог rf() в R.

```
bool  MathRandomF(
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsF

Рассчитывает теоретические численные значения первых 4 моментов F-распределения Фишера с параметрами nu1 и nu2.

```
double  MathMomentsF(
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

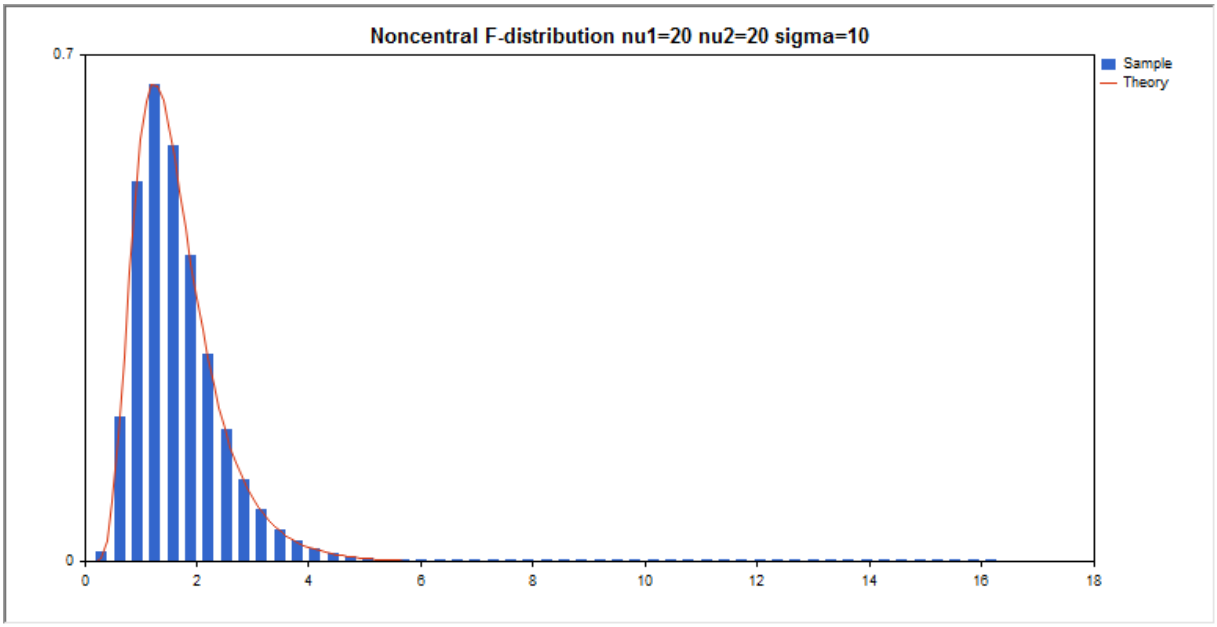
Нецентральное F-распределение

В данном разделе представлены функции для работы с нецентральным F-распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по закону нецентрального распределения Фишера. Нецентральное F-распределение описывается следующей формулой:

$$f_{NoncentralF}(x|\nu_1,\nu_2,\sigma) = e^{-\frac{\sigma}{2}} \sum_{r=0}^{\infty} \frac{1}{r!}\left(\frac{\sigma}{2}\right)^r \frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}+r\right)}{\Gamma\left(\frac{\nu_2}{2}+r\right)\Gamma\left(\frac{\nu_2}{2}\right)} \left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_2}{2}+r} \frac{x^{\frac{\nu_2}{2}-1+r}}{\left(1+\frac{\nu_1}{\nu_2}x\right)^{\frac{\nu_1+\nu_2}{2}+r}}$$

где:

- x — значение случайной величины
- ν1 — первый параметр распределения (число степеней свободы)
- ν2 — второй параметр распределения (число степеней свободы)
- σ — параметр нецентральности



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityNoncentralF	Рассчитывает плотность вероятности нецентрального F-распределения
MathCumulativeDistributionNoncentralF	Рассчитывает значение функции нецентрального F-распределения вероятностей
MathQuantileNoncentralF	Рассчитывает значение обратной функции нецентрального F-распределения для заданной вероятности

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathRandomNoncentralF	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону нецентрального распределения Фишера
MathMomentsNoncentralF	Рассчитывает теоретические численные значения первых 4 моментов нецентрального распределения Фишера

Пример:

```
#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;     // первое число степеней свободы
input double nu_2=20;     // второе число степеней свободы
input double sig=10;      // параметр нецентральности
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // количество значений в выборке
   int ncells=51;        // количество интервалов в гистограмме
   double x[];           // центры интервалов гистограммы
   double y[];           // количество значений из выборки, попавших в интервал
   double data[];        // выборка случайных значений
   double max,min;       // максимальное и минимальное значения в выборке
//--- получим выборку из нецентрального F-распределения
   MathRandomNoncentralF(nu_1,nu_2,sig,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
```

© 2000-2025, MetaQuotes Ltd.

```
   MathProbabilityDensityNoncentralF(x2,nu_1,nu_2,sig,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
```

© 2000-2025, MetaQuotes Ltd.

```
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralF

Рассчитывает плотность вероятности нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // значение случайной величины
   const double  nu1,            // первый параметр распределения (число степеней свобо
   const double  nu2,            // второй параметр распределения (число степеней свобо
   const double  sigma,          // параметр нецентральности
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // значение случайной величины
   const double  nu1,            // первый параметр распределения (число степеней свобо
   const double  nu2,            // второй параметр распределения (число степеней свобо
   const double  sigma,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для массива случайных величин x[]. В случае ошибки возвращает false. Аналог df() в R.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // массив со значениями случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // массив со значениями случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

© 2000-2025, MetaQuotes Ltd.

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralF

Рассчитывает распределение вероятностей по закону нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // значение случайной величины
   const double  nu1,            // первый параметр распределения (число степеней свобо
   const double  nu2,            // второй параметр распределения (число степеней свобо
   const double  sigma,          // параметр нецентральности
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей по закону нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // значение случайной величины
   const double  nu1,            // первый параметр распределения (число степеней свобо
   const double  nu2,            // второй параметр распределения (число степеней свобо
   const double  sigma,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей по закону нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pf() в R.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // массив со значениями случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает распределение вероятностей по закону нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // массив со значениями случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   double&       result[]        // массив для значений функции вероятности
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileF

Рассчитывает значение обратной функции нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileF(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileF(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qf() в R.

```
double  MathQuantileF(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileF(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   double&       result[]        // массив со значениями квантилей
   );
```

© 2000-2025, MetaQuotes Ltd.

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralF

Генерирует псевдослучайную величину, распределенную по закону нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma. В случае ошибки возвращает NaN.

```
double  MathRandomNoncentralF(
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, по закону нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma. В случае ошибки возвращает false. Аналог rf() в R.

```
bool  MathRandomNoncentralF(
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralF

Рассчитывает теоретические численные значения первых 4 моментов нецентрального F-распределения Фишера с параметрами nu1, nu2 и sigma.

```
double  MathMomentsNoncentralF(
   const double  nu1,            // первый параметр распределения (число степеней своб
   const double  nu2,            // второй параметр распределения (число степеней своб
   const double  sigma,          // параметр нецентральности
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

nu1

[in] Первый параметр распределения (число степеней свободы).

nu2

[in] Второй параметр распределения (число степеней свободы).

sigma

[in] Параметр нецентральности.

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

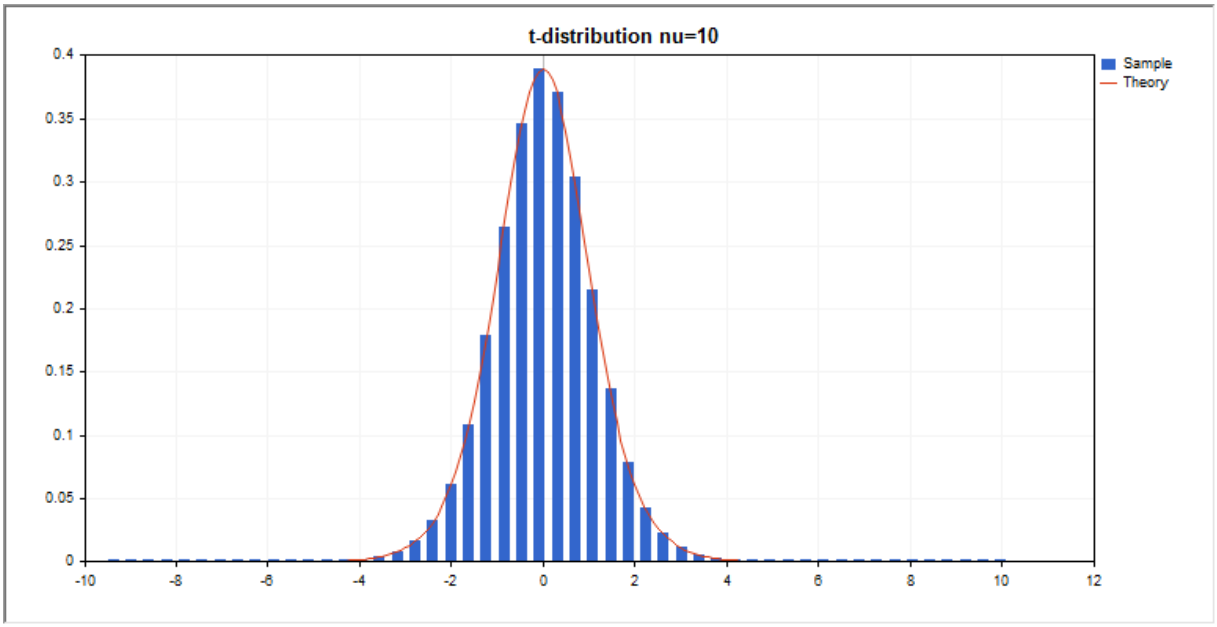
T-распределение

В данном разделе представлены функции для работы с T-распределением Стьюдента. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по закону Стьюдента. T-распределение описывается следующей формулой:

$$f_T(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\pi\nu}} \frac{1}{\left(1+\frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}}$$

где:

- x — значение случайной величины
- ν — параметр распределения (число степеней свободы)



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityT	Рассчитывает плотность вероятности T-распределения
MathCumulativeDistributionT	Рассчитывает значение функции T-распределения вероятностей
MathQuantileT	Рассчитывает значение обратной функции T-распределения для заданной вероятности
MathRandomT	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону Стьюдента

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathMomentsT	Рассчитывает теоретические численные значения первых 4 моментов T-распределения Стьюдента

Пример:

```
#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;     // число степеней свободы
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // количество значений в выборке
   int ncells=51;          // количество интервалов в гистограмме
   double x[];             // центры интервалов гистограммы
   double y[];             // количество значений из выборки, попавших в интервал
   double data[];          // выборка случайных значений
   double max,min;         // максимальное и минимальное значения в выборке
//--- получим выборку из T-распределения
   MathRandomT(nu_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityT(x2,nu_par,false,y2);
//--- масштабируем
   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++)
```

© 2000-2025, MetaQuotes Ltd.

```
      y[i]/=k;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityT

Рассчитывает плотность вероятности T-распределения Стьюдента с параметром nu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityT(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности T-распределения Стьюдента с параметром nu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityT(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности T-распределения Стьюдента с параметром nu для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dt() в R.

```
bool  MathProbabilityDensityT(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности T-распределения Стьюдента с параметром nu для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityT(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionT

Рассчитывает распределение Стьюдента с параметром nu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionT(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение Стьюдента с параметром nu для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionT(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение Стьюдента с параметром nu для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pt() в R.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает распределение Стьюдента с параметром nu для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileT

Рассчитывает значение обратной функции T-распределения Стьюдента с параметром nu для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileT(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции T-распределения Стьюдента с параметром nu для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileT(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции T-распределения Стьюдента с параметром nu для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qt() в R.

```
double  MathQuantileT(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  nu,             // параметр распределения (число степеней свободы)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции T-распределения Стьюдента с параметром nu для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileT(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  nu,             // параметр распределения (число степеней свободы)
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomT

Генерирует псевдослучайную величину, распределенную по закону T-распределения Стьюдента с параметром nu. В случае ошибки возвращает NaN.

```
double  MathRandomT(
   const double  nu,              // параметр распределения (число степеней свободы)
   int&          error_code       // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону T-распределения Стьюдента с параметром nu. В случае ошибки возвращает false. Аналог rt() в R.

```
bool  MathRandomT(
   const double  nu,              // параметр распределения (число степеней свободы)
   const int     data_count,      // количество необходимых данных
   double&       result[]         // массив со значениями псевдослучайных величин
   );
```

Параметры

nu

[in] Параметр распределения (число степеней свободы).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsT

Рассчитывает теоретические численные значения первых 4 моментов T-распределения Стьюдента с параметром nu.

```
double  MathMomentsT(
   const double  nu,              // параметр распределения (число степеней свободы)
   double&       mean,            // переменная для среднего значения
   double&       variance,        // переменная для дисперсии
   double&       skewness,        // переменная для коэффициента асимметрии
   double&       kurtosis,        // переменная для коэффициента эксцесса
   int&          error_code       // переменная для кода ошибки
   );
```

Параметры

nu

[in] Параметр распределения (число степеней свободы).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

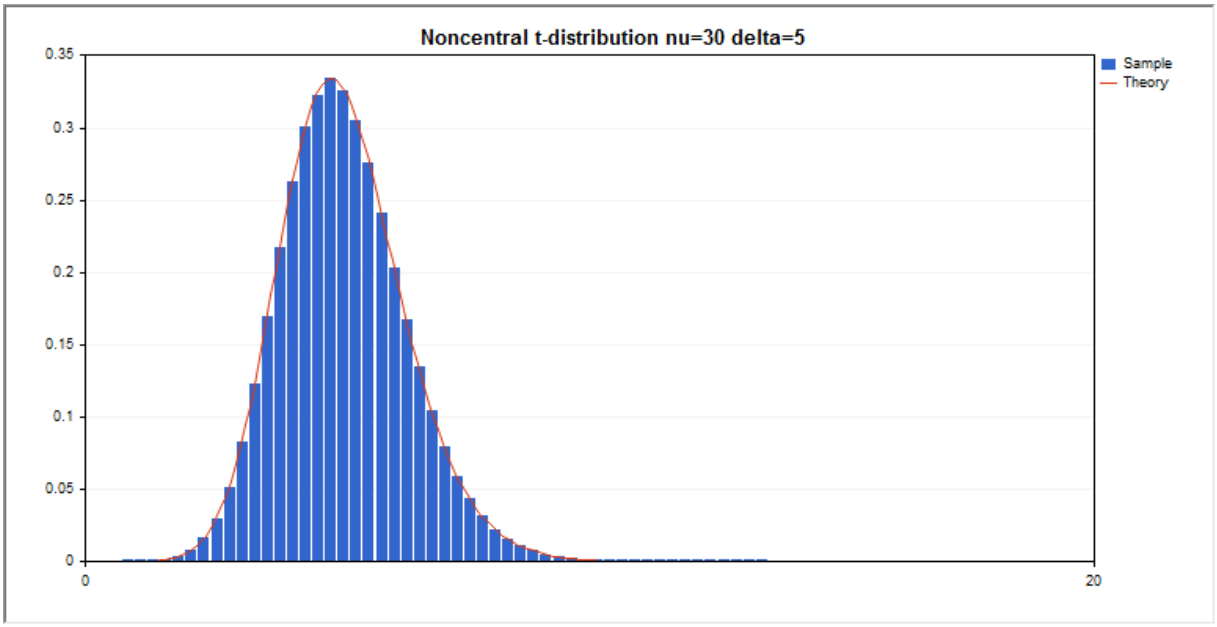
T-распределение

В данном разделе представлены функции для работы с нецентральным T-распределением Стьюдента. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по закону нецентрального T-распределения. Нецентральное T-распределение описывается следующей формулой:

$$f_{NoncentralT}(x|\nu,\delta)=\frac{\nu^{\frac{\nu}{2}}e^{-\frac{\sigma^2}{2}}}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi}\left(\nu+x^2\right)^{\frac{\nu+1}{2}}}\sum_{r=0}^{\infty}\frac{(x\delta)^r}{r!}\left(\frac{2}{\nu+x^2}\right)^{\frac{r}{2}}\Gamma\left(\frac{\nu+r+1}{2}\right)$$

где:

- x — значение случайной величины
- ν — параметр распределения (число степеней свободы)
- σ — параметр нецентральности



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityNoncentralT	Рассчитывает плотность вероятности нецентрального T-распределения
MathCumulativeDistributionNoncentralT	Рассчитывает значение функции нецентрального T-распределения вероятностей
MathQuantileNoncentralT	Рассчитывает значение обратной функции нецентрального T-распределения для заданной вероятности
MathRandomNoncentralT	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону нецентрального T-распределения Стьюдента

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathMomentsNoncentralT	Рассчитывает теоретические численные значения первых 4 моментов нецентрального T-распределения Стьюдента

Пример:

```
#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;        // число степеней свободы
input double delta_par=5;     // параметр нецентральности
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // количество значений в выборке
   int ncells=51;        // количество интервалов в гистограмме
   double x[];           // центры интервалов гистограммы
   double y[];           // количество значений из выборки, попавших в интервал
   double data[];        // выборка случайных значений
   double max,min;       // максимальное и минимальное значения в выборке
//--- получим выборку из нецентрального T-распределения
   MathRandomNoncentralT(nu_par,delta_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralT(x2,nu_par,delta_par,false,y2);
//--- масштабируем
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- выводим графики
   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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralT

Рассчитывает плотность вероятности нецентрального T-распределения Стьюдента с параметрами nu и delta для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для кода ошибки
   );
```

Рассчитывает плотность вероятности нецентрального T-распределения Стьюдента с параметрами nu и delta для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   int&          error_code      // переменная для кода ошибки
   );
```

Рассчитывает плотность вероятности нецентрального T-распределения Стьюдента с параметрами nu и delta для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dt() в R.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности нецентрального T-распределения Стьюдента с параметрами nu и delta для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

nu

[in] Параметр распределения (число степеней свободы).

delta

[in] Параметр нецентральности.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralT

Рассчитывает распределение вероятностей нецентрального T-распределения с параметрами nu и delta для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей нецентрального T-распределения с параметрами nu и delta для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // значение случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей нецентрального T-распределения с параметрами nu и delta для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pt() в R.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает распределение вероятностей нецентрального T-распределения с параметрами nu и delta для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // массив со значениями случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

delta

[in] Параметр нецентральности.

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralT

Рассчитывает значение обратной функции нецентрального T-распределения Стьюдента с параметрами nu и delta для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   const bool    tail,           // флаг расчета, если lower_tail=false, то расчет вед
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции нецентрального T-распределения Стьюдента с параметрами nu и delta для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // значение вероятности появления случайной величины
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции нецентрального T-распределения Стьюдента с параметрами nu и delta для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qt() в R.

```
double  MathQuantileNoncentralT(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   const bool    tail,           // флаг расчета, если lower_tail=false, то расчет вед
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции нецентрального T-распределения Стьюдента с параметрами nu и delta для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileNoncentralT(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

© 2000-2025, MetaQuotes Ltd.

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

nu

[in] Параметр распределения (число степеней свободы).

delta

[in] Параметр нецентральности.

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralT

Генерирует псевдослучайную величину, распределенную по закону нецентрального T-распределения Стьюдента с параметрами nu и delta. В случае ошибки возвращает NaN.

```
double  MathRandomNoncentralT(
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону нецентрального T-распределения Стьюдента с параметрами nu и delta. В случае ошибки возвращает false. Аналог rt() в R.

```
bool  MathRandomNoncentralT(
   const double  nu,             // параметр распределения (число степеней свободы)
   const double  delta,          // параметр нецентральности
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

nu

[in] Параметр распределения (число степеней свободы).

delta

[in] Параметр нецентральности.

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralT

Рассчитывает теоретические численные значения первых 4 моментов нецентрального T-распределения Стьюдента с параметрами nu и delta.

```
double  MathMomentsNoncentralT(
   const double  nu,              // параметр распределения (число степеней свободы)
   const double  delta,           // параметр нецентральности
   double&       mean,            // переменная для среднего значения
   double&       variance,        // переменная для дисперсии
   double&       skewness,        // переменная для коэффициента асимметрии
   double&       kurtosis,        // переменная для коэффициента эксцесса
   int&          error_code       // переменная для кода ошибки
   );
```

Параметры

nu

[in] Параметр распределения (число степеней свободы).

delta

[in] Параметр нецентральности.

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

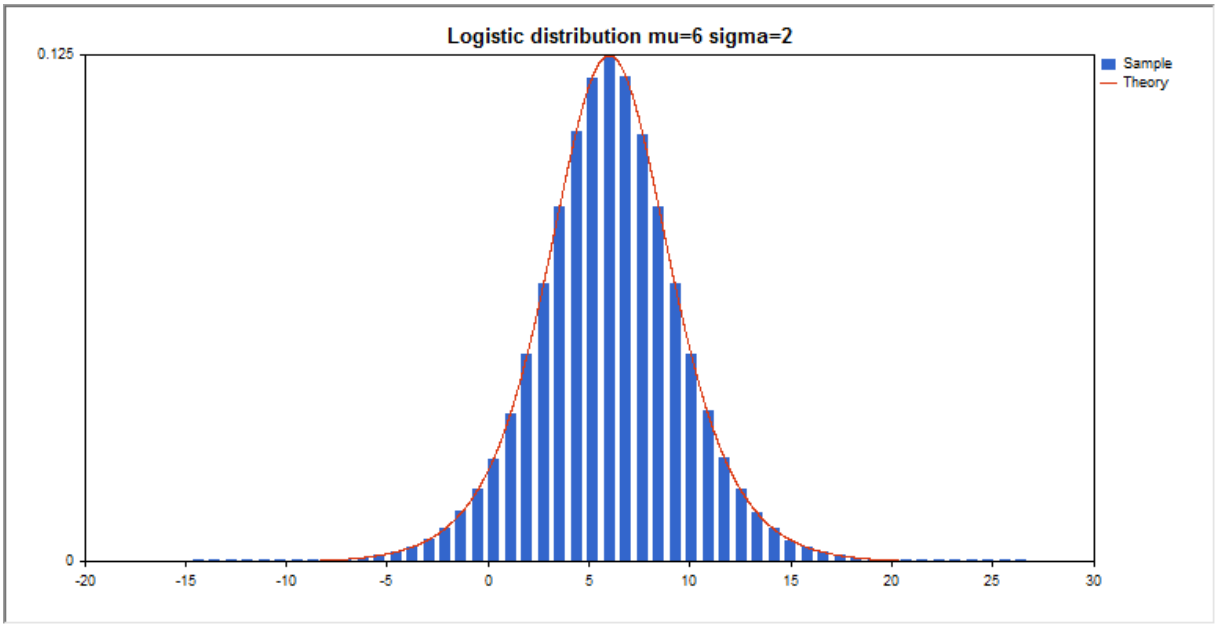
Логистическое распределение

В данном разделе представлены функции для работы с логистическим распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по логистическому закону. Логистическое распределение описывается следующей формулой:

$$f_{Logistic}(x \mid \mu, \sigma) = \frac{e^{\frac{x-\mu}{\sigma}}}{\sigma\left(1+e^{\frac{x-\mu}{\sigma}}\right)^2}$$

где:

- x — значение случайной величины
- μ — параметр распределения mean
- σ — параметр распределения scale



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityLogistic	Рассчитывает плотность вероятности логистического распределения
MathCumulativeDistributionLogistic	Рассчитывает значение функции логистического распределения вероятностей
MathQuantileLogistic	Рассчитывает значение обратной функции логистического распределения для заданной вероятности

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathRandomLogistic	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по логистическому закону
MathMomentsLogistic	Рассчитывает теоретические численные значения первых 4 моментов логистического распределения

Пример:

```
#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;         // параметр распределения mean
input double sigma_par=2;      // параметр распределения scale
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // количество значений в выборке
   int ncells=51;          // количество интервалов в гистограмме
   double x[];             // центры интервалов гистограммы
   double y[];             // количество значений из выборки, попавших в интервал
   double data[];          // выборка случайных значений
   double max,min;         // максимальное и минимальное значения в выборке
//--- получим выборку из логистического распределения
   MathRandomLogistic(mu_par,sigma_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLogistic(x2,mu_par,sigma_par,false,y2);
//--- масштабируем
```

© 2000-2025, MetaQuotes Ltd.

```
   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;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      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);
//--- отключим автомасштабирование оси 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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       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)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityLogistic

Рассчитывает плотность вероятности логистического распределения с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности логистического распределения с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности логистического распределения с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dlogis() в R.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности логистического распределения с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

mu

[in] Параметр распределения mean.

sigma

[in] Параметр распределения scale.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionLogistic

Рассчитывает логистическое распределение вероятностей с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает логистическое распределение вероятностей с параметрами mu и sigma для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // значение случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает логистическое распределение вероятностей с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает логистическое распределение вероятностей с параметрами mu и sigma для массива случайных величин x[]. В случае ошибки возвращает false. Аналог plogis() в R.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // массив со значениями случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

mu

[in] Параметр распределения mean.

sigma

[in] Параметр распределения scale.

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileLogistic

Рассчитывает значение обратной функции логистического распределения с параметрами mu и sigma для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // значение вероятности появления случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции логистического распределения с с параметрами mu и sigma для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // значение вероятности появления случайной величины
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции логистического распределения с с параметрами mu и sigma для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qlogis() в R.

```
double  MathQuantileLogistic(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции логистического распределения с с параметрами mu и sigma для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileLogistic(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  mu,             // параметр распределения mean
   const double  sigma,          // параметр распределения scale
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Массив со значениями вероятностей случайной величины.

mu

[in] Параметр распределения mean.

sigma

[in] Параметр распределения scale.

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomLogistic

Генерирует псевдослучайную величину, распределенную по закону логистического распределения с параметрами mu и sigma. В случае ошибки возвращает NaN.

```
double  MathRandomLogistic(
   const double  mu,              // параметр распределения mean
   const double  sigma,           // параметр распределения scale
   int&          error_code       // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону логистического распределения с параметрами mu и sigma. В случае ошибки возвращает false. Аналог rlogis() в R.

```
bool  MathRandomLogistic(
   const double  mu,              // параметр распределения mean
   const double  sigma,           // параметр распределения scale
   const int     data_count,      // количество необходимых данных
   double&       result[]         // массив со значениями псевдослучайных величин
   );
```

Параметры

mu

[in] Параметр распределения mean.

sigma

[in] Параметр распределения scale.

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsLogistic

Рассчитывает теоретические численные значения первых 4 моментов логистического распределения с параметрами mu и sigma.

```
double  MathMomentsLogistic(
   const double  mu,              // параметр распределения mean
   const double  sigma,           // параметр распределения scale
   double&       mean,            // переменная для среднего значения
   double&       variance,        // переменная для дисперсии
   double&       skewness,        // переменная для коэффициента асимметрии
   double&       kurtosis,        // переменная для коэффициента эксцесса
   int&          error_code       // переменная для кода ошибки
   );
```

Параметры

mu

[in] Параметр распределения mean.

sigma

[in] Параметр распределения scale.

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

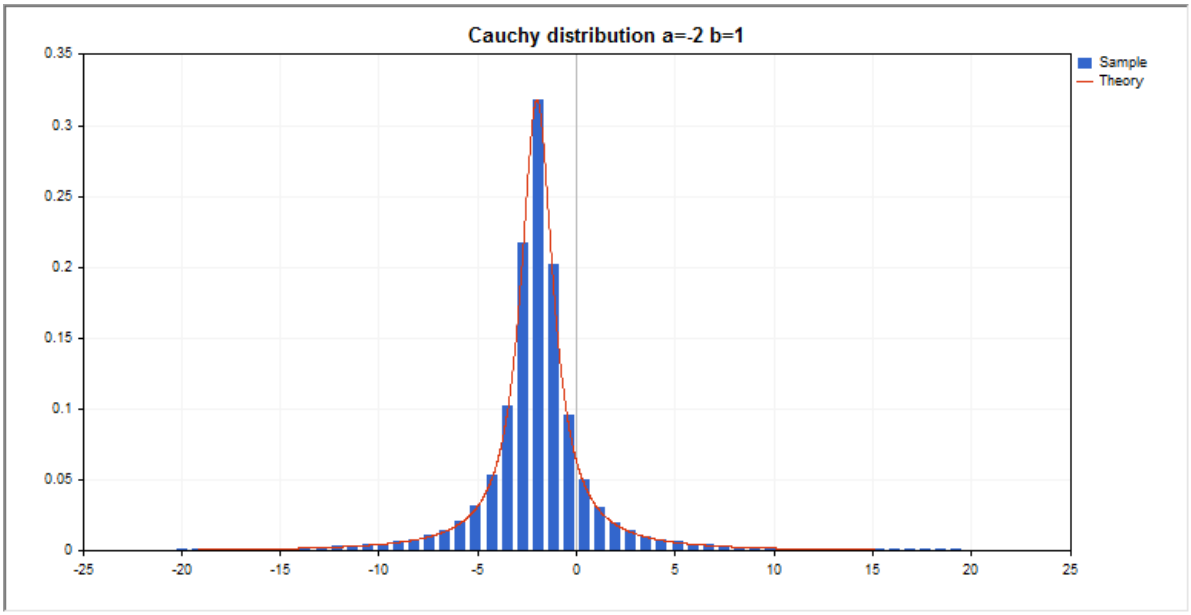
Распределение Коши

В данном разделе представлены функции для работы с распределением Коши. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по закону Коши. Распределение Коши описывается следующей формулой:

$$f_{Cauchy}(x|a,b) = \frac{1}{\pi}\frac{b}{\left(b^2+(x-a)^2\right)}$$

где:

- x — значение случайной величины
- a — параметр распределения mean
- b — параметр распределения scale



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityCauchy	Рассчитывает плотность вероятности распределения Коши
MathCumulativeDistributionCauchy	Рассчитывает значение функции распределения вероятностей Коши
MathQuantileCauchy	Рассчитывает значение обратной функции распределения Коши для заданной вероятности
MathRandomCauchy	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону Коши
MathMomentsCauchy	Рассчитывает теоретические численные значения первых 4 моментов распределения Коши

© 2000-2025, MetaQuotes Ltd.

Пример:

```
#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;      // параметр распределения mean
input double b_par=1;       // параметр распределения scale
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // количество значений в выборке
   int ncells=51;          // количество интервалов в гистограмме
   double x[];             // центры интервалов гистограммы
   double y[];             // количество значений из выборки, попавших в интервал
   double data[];          // выборка случайных значений
   double max,min;         // максимальное и минимальное значения в выборке
//--- получим выборку из распределения Коши
   MathRandomCauchy(a_par,b_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityCauchy(x2,a_par,b_par,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=(int)MathRound((data[i]-minv)/width);
      if(ind>=0 && ind<cells)
         frequency[ind]++;
     }
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityCauchy

Рассчитывает плотность вероятности распределения Коши с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности распределения Коши с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности распределения Коши с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dcauchy() в R.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности распределения Коши с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

a

[in] Параметр распределения mean.

b

[in] Параметр распределения scale.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionCauchy

Рассчитывает распределение вероятностей Коши с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей Коши с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение вероятностей Коши с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает распределение вероятностей Коши с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог plogis() в R.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

a

[in] Параметр распределения mean.

b

[in] Параметр распределения scale.

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileCauchy

Рассчитывает значение обратной функции распределения Коши с параметрами a и b для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции распределения Коши с параметрами a и b для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции распределения Коши с параметрами a и b для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qcauschy() в R.

```
double  MathQuantileCauchy(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции распределения Коши с параметрами a и b для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileCauchy(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями вероятностей случайной величины.

a

[in] Параметр распределения mean.

b

[in] Параметр распределения scale.

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomCauchy

Генерирует псевдослучайную величину, распределенную по закону распределения Коши с параметрами a и b. В случае ошибки возвращает NaN.

```
double  MathRandomCauchy(
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону распределения Коши с параметрами a и b. В случае ошибки возвращает false. Аналог rcauchy() в R.

```
bool  MathRandomCauchy(
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

a

[in] Параметр распределения mean.

b

[in] Параметр распределения scale.

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsCauchy

Рассчитывает теоретические численные значения первых 4 моментов распределения Коши с параметрами a и b.

```
double  MathMomentsCauchy(
   const double  a,              // параметр распределения mean
   const double  b,              // параметр распределения scale
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

a

[in] Параметр распределения mean.

b

[in] Параметр распределения scale.

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

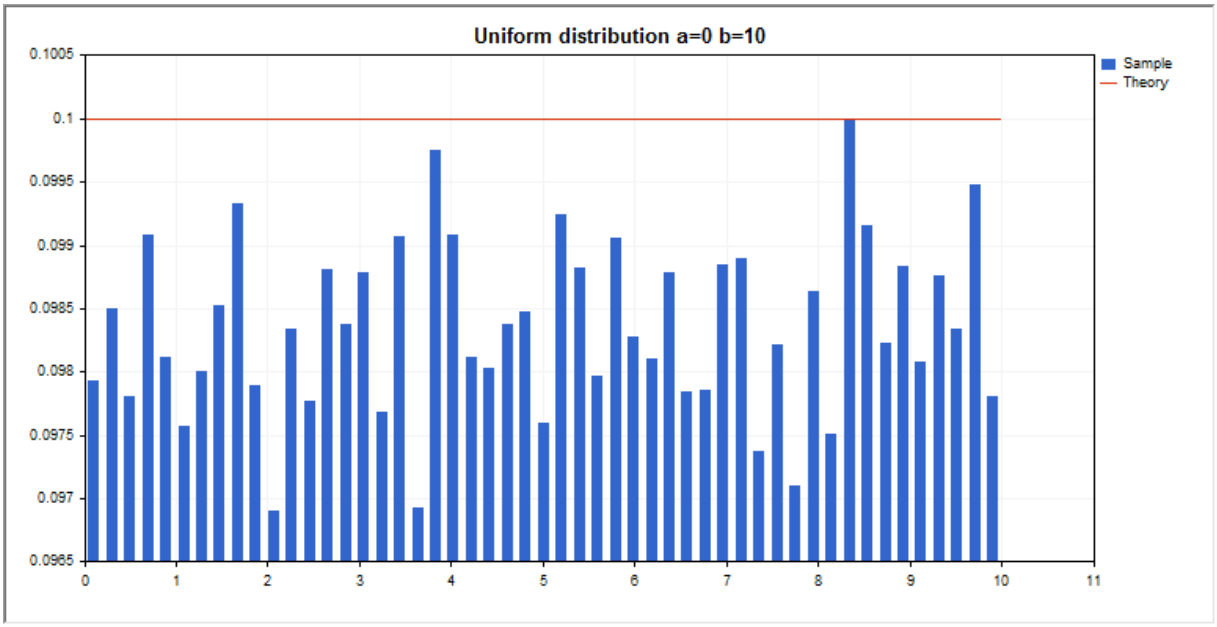
Равномерное распределение

В данном разделе представлены функции для работы с равномерным распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по равномерному закону. Равномерное распределение описывается следующей формулой:

$$f_{Uniform}(x|\ a,b) = \frac{1}{b-a}$$

где:

- x — значение случайной величины
- a — параметр распределения (нижняя граница)
- b — параметр распределения (верхняя граница)



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityUniform	Рассчитывает плотность вероятности равномерного распределения
MathCumulativeDistributionUniform	Рассчитывает значение функции равномерного распределения вероятностей
MathQuantileUniform	Рассчитывает значение обратной функции равномерного распределения для заданной вероятности
MathRandomUniform	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по равномерному закону
MathMomentsUniform	Рассчитывает теоретические численные значения первых 4 моментов равномерного распределения

© 2000-2025, MetaQuotes Ltd.

Пример:

```
#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;      // параметр распределения a (нижняя граница)
input double b_par=10;     // параметр распределения b (верхняя граница)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // количество значений в выборке
   int ncells=51;          // количество интервалов в гистограмме
   double x[];             // центры интервалов гистограммы
   double y[];             // количество значений из выборки, попавших в интервал
   double data[];          // выборка случайных значений
   double max,min;         // максимальное и минимальное значения в выборке
//--- получим выборку из равномерного распределения
   MathRandomUniform(a_par,b_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityUniform(x2,a_par,b_par,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   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)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityUniform

Рассчитывает плотность вероятности равномерного распределения с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности равномерного распределения с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности равномерного распределения с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dunif() в R.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности равномерного распределения с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

a

[in] Параметр распределения a (нижняя граница).

b

[in] Параметр распределения b (верхняя граница).

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionUniform

Рассчитывает равномерное распределение вероятностей с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает равномерное распределение вероятностей с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает равномерное распределение вероятностей с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает равномерное распределение вероятностей с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог punif() в R.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

a

[in] Параметр распределения a (нижняя граница).

b

[in] Параметр распределения b (верхняя граница).

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileUniform

Рассчитывает значение обратной функции равномерного распределения с параметрами a и b для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileUniform(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции равномерного распределения с параметрами a и b для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileUniform(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции равномерного распределения с параметрами a и b для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qcauschy() в R.

```
double  MathQuantileUniform(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции равномерного распределения с параметрами a и b для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileUniform(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Массив со значениями вероятностей случайной величины.

a

[in] Параметр распределения a (нижняя граница).

b

[in] Параметр распределения b (верхняя граница).

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomUniform

Генерирует псевдослучайную величину, распределенную по закону равномерного распределения с параметрами a и b. В случае ошибки возвращает NaN.

```
double  MathRandomUniform(
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону равномерного распределения с параметрами a и b. В случае ошибки возвращает false. Аналог runif() в R.

```
bool  MathRandomUniform(
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

a

[in] Параметр распределения a (нижняя граница).

b

[in] Параметр распределения b (верхняя граница).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsUniform

Рассчитывает теоретические численные значения первых 4 моментов равномерного распределения с параметрами a и b.

```
double  MathMomentsUniform(
   const double  a,              // параметр распределения a (нижняя граница)
   const double  b,              // параметр распределения b (верхняя граница)
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

a

[in] Параметр распределения a (нижняя граница).

b

[in] Параметр распределения b (верхняя граница).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

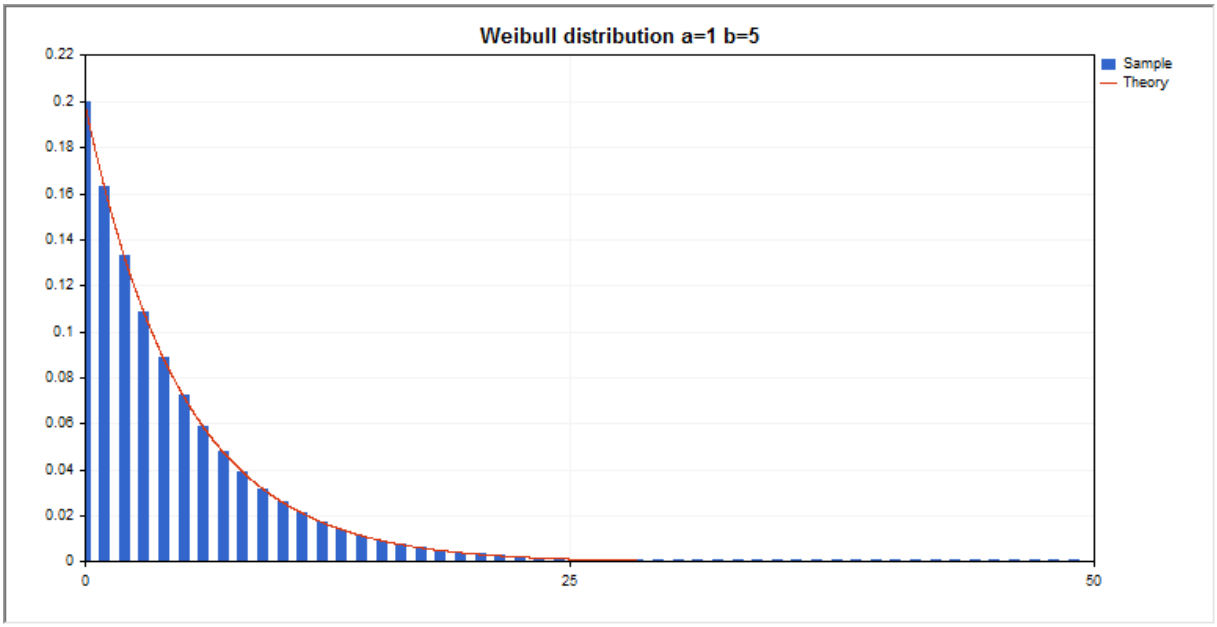
Распределение Вейбулла

В данном разделе представлены функции для работы с распределением Вейбулла. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по закону Вейбулла. Распределение Вейбулла описывается следующей формулой:

$$f_{Weibull}(x|a,b) = \frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-\left(\frac{x}{b}\right)^{a}}$$

где:

- x — значение случайной величины
- a — параметр распределения (shape)
- b — параметр распределения (scale)



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityWeibull	Рассчитывает плотность вероятности распределения Вейбулла
MathCumulativeDistributionWeibull	Рассчитывает значение функции распределения вероятностей Вейбулла
MathQuantileWeibull	Рассчитывает значение обратной функции распределения Вейбулла для заданной вероятности
MathRandomWeibull	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону Вейбулла
MathMomentsWeibull	Рассчитывает теоретические численные значения первых 4 моментов распределения Вейбулла

© 2000-2025, MetaQuotes Ltd.

Пример:

```
#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;      // параметр распределения (shape)
input double b_par=5;      // параметр распределения (scale)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // количество значений в выборке
   int ncells=51;          // количество интервалов в гистограмме
   double x[];             // центры интервалов гистограммы
   double y[];             // количество значений из выборки, попавших в интервал
   double data[];          // выборка случайных значений
   double max,min;         // максимальное и минимальное значения в выборке
//--- получим выборку из распределения Вейбулла
   MathRandomWeibull(a_par,b_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityWeibull(x2,a_par,b_par,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, 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);
//--- отключим автомасштабирование оси 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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityWeibull

Рассчитывает плотность вероятности распределения Вейбулла с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности распределения Вейбулла с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает плотность вероятности распределения Вейбулла с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dweibull() в R.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает плотность вероятности распределения Вейбулла с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

© 2000-2025, MetaQuotes Ltd.

a

[in] Параметр распределения (scale).

b

[in] Параметр распределения (shape).

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionWeibull

Рассчитывает распределение Вейбулла с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение Вейбулла с параметрами a и b для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // значение случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает распределение Вейбулла с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pweibull() в R.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает распределение Вейбулла с параметрами a и b для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // массив со значениями случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

a

[in] Параметр распределения (scale).

b

[in] Параметр распределения (shape).

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileWeibull

Рассчитывает значение обратной функции распределения вероятностей Вейбулла с параметрами a и b для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции распределения вероятностей Вейбулла с параметрами a и b для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // значение вероятности появления случайной величины
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение обратной функции распределения вероятностей Вейбулла с параметрами a и b для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qweibull() в R.

```
double  MathQuantileWeibull(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает значение обратной функции распределения вероятностей Вейбулла с параметрами a и b для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileWeibull(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

© 2000-2025, MetaQuotes Ltd.

probability[]

[in] Массив со значениями вероятностей случайной величины.

a

[in] Параметр распределения (scale).

b

[in] Параметр распределения (shape).

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomWeibull

Генерирует псевдослучайную величину, распределенную по закону распределения Вейбулла с параметрами a и b. В случае ошибки возвращает NaN.

```
double  MathRandomWeibull(
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону распределения Вейбулла с параметрами a и b. В случае ошибки возвращает false. Аналог rweibull() в R.

```
bool  MathRandomWeibull(
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

a

[in] Параметр распределения (scale).

b

[in] Параметр распределения (shape).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsWeibull

Рассчитывает теоретические численные значения первых 4 моментов распределения Вейбулла с параметрами a и b.

```
double  MathMomentsWeibull(
   const double  a,              // параметр распределения (shape)
   const double  b,              // параметр распределения (scale)
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

a

[in] Параметр распределения (scale).

b

[in] Параметр распределения (shape).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

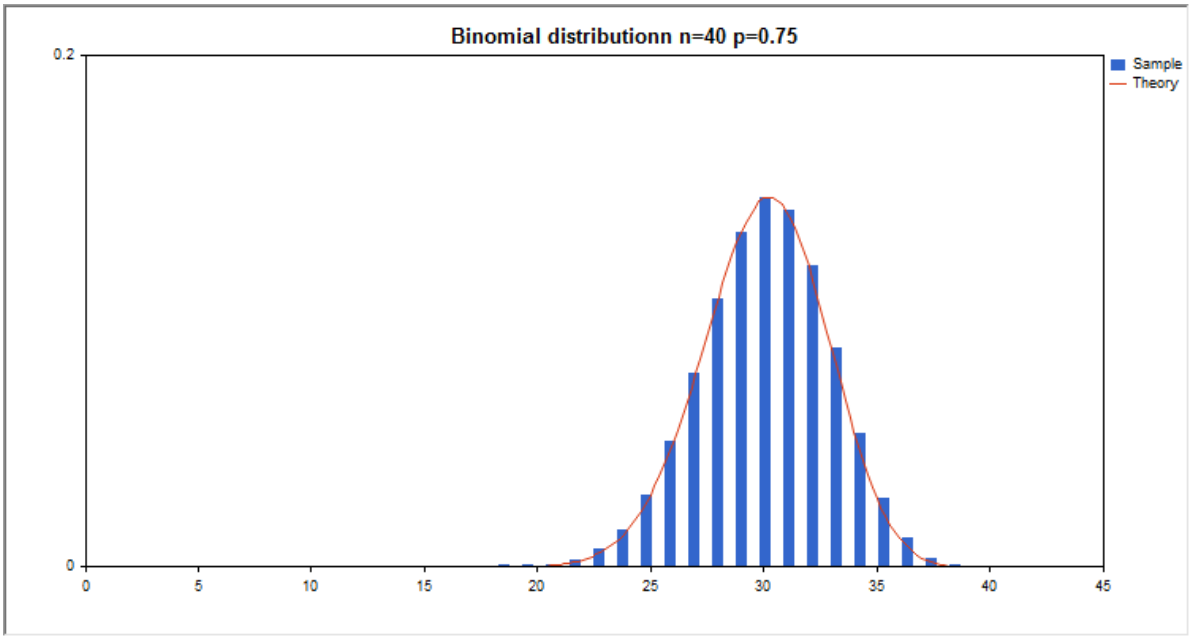
Биномиальное распределение

В данном разделе представлены функции для работы с биномиальным распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по биномиальному закону. Биномиальное распределение описывается следующей формулой:

$$f_{Binomial}(x \mid n, p) = \binom{n}{x} p^x (1-p)^{n-x}$$

где:

- x — значение случайной величины
- n — количество испытаний
- p — вероятность успеха для каждого испытания



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityBinomial	Рассчитывает плотность вероятности биномиального распределения
MathCumulativeDistributionBinomial	Рассчитывает значение функции биномиального распределения вероятностей
MathQuantileBinomial	Рассчитывает значение обратной функции биномиального распределения для заданной вероятности
MathRandomBinomial	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по биномиальному закону

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathMomentsBinomial	Рассчитывает теоретические численные значения первых 4 моментов биномиального распределения

Пример:

```
#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;         // количество испытаний
input double p_par=0.75;       // вероятность успеха для каждого испытания
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;         // количество значений в выборке
   int ncells=20;         // количество интервалов в гистограмме
   double x[];            // центры интервалов гистограммы
   double y[];            // количество значений из выборки, попавших в интервал
   double data[];         // выборка случайных значений
   double max,min;        // максимальное и минимальное значения в выборке
//--- получим выборку из биномиального распределения
   MathRandomBinomial(n_par,p_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityBinomial(x2,n_par,p_par,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   CGraphic graphic;
```

© 2000-2025, MetaQuotes Ltd.

```
   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)
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityBinomial

Рассчитывает значение функции вероятности (probability mass function) биномиального распределения с параметрами n и p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // значение случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для кажд
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) биномиального распределения с параметрами n и p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // значение случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для кажд
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) биномиального распределения с параметрами n и p для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dbinom() в R.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // массив со значениями случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для как
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает значение функции вероятности (probability mass function) биномиального распределения с параметрами n и p для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // массив со значениями случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для как
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

n

[in] Параметр распределения (количество испытаний).

p

[in] Параметр распределения (вероятность успеха для каждого испытания).

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionBinomial

Рассчитывает функцию распределения для биномиального закона с параметрами n и p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // значение случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для кажд
   const bool    tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает функцию распределения для биномиального закона с параметрами n и p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // значение случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для кажд
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает функцию распределения для биномиального закона с параметрами n и p для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pweibull() в R.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // массив со значениями случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для как
   const bool    tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает функцию распределения для биномиального закона с параметрами n и p для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // массив со значениями случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для как
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

n

[in] Параметр распределения (количество испытаний).

p

[in] Параметр распределения (вероятность успеха для каждого испытания).

tail

[in] Флаг расчета. Если true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileBinomial

Рассчитывает обратное значение функции распределения для биномиального закона с параметрами n и p для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // значение вероятности появления случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для ка
   const bool    tail,           // флаг расчета, если false, то расчет ведется для в
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведет
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения для биномиального закона с параметрами n и p для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // значение вероятности появления случайной величины
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для ка
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения для биномиального закона с параметрами n и p для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qbinom() в R.

```
double  MathQuantileBinomial(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для ка
   const bool    tail,           // флаг расчета, если false, то расчет ведется для в
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведет
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает обратное значение функции распределения для биномиального закона с параметрами n и p для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileBinomial(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для ка
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

© 2000-2025, MetaQuotes Ltd.

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

n

[in] Параметр распределения (количество испытаний).

p

[in] Параметр распределения (вероятность успеха для каждого испытания).

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomBinomial

Генерирует псевдослучайную величину, распределенную по закону биномиального распределения с параметрами n и p. В случае ошибки возвращает NaN.

```
double  MathRandomBinomial(
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для ка
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону биномиального распределения с параметрами n и p. В случае ошибки возвращает false. Аналог rweibull() в R.

```
bool  MathRandomBinomial(
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для ка
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

n

[in] Параметр распределения (количество испытаний).

p

[in] Параметр распределения (вероятность успеха для каждого испытания).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsBinomial

Рассчитывает теоретические численные значения первых 4 моментов биномиального распределения с параметрами n и p.

```
double  MathMomentsBinomial(
   const double  n,              // параметр распределения (количество испытаний)
   const double  p,              // параметр распределения (вероятность успеха для ка
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

n

[in] Параметр распределения (количество испытаний).

p

[in] Параметр распределения (вероятность успеха для каждого испытания).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

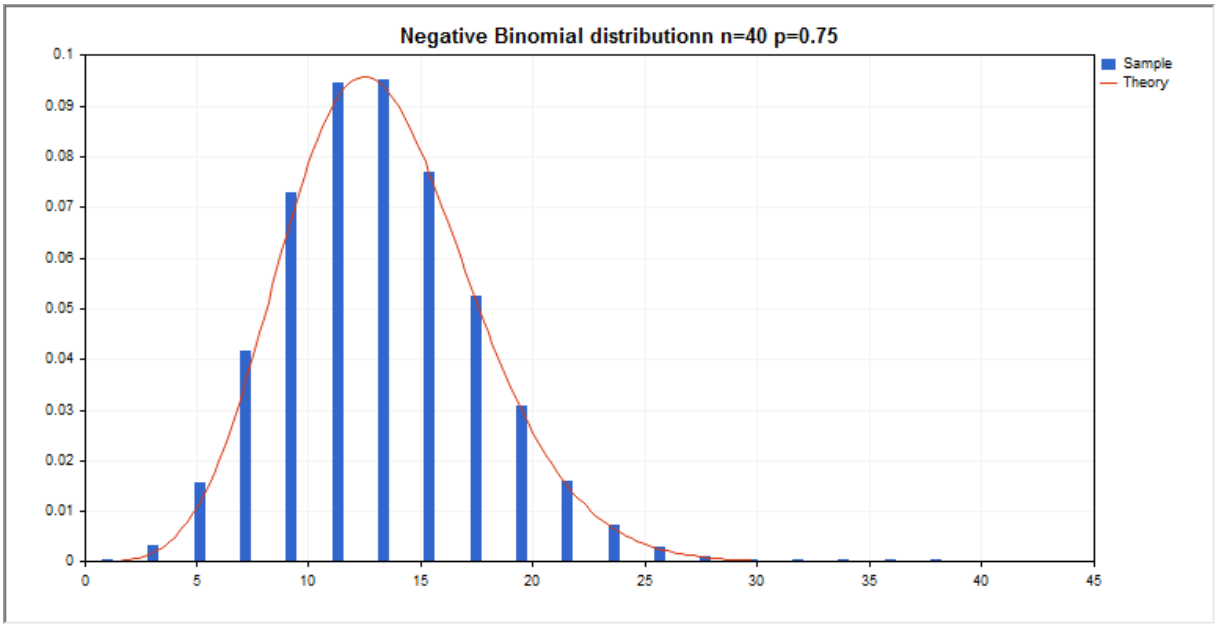
Отрицательное биномиальное распределение

В данном разделе представлены функции для работы с отрицательным биномиальным распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по отрицательному биномиальному закону. Отрицательное биномиальное распределение описывается следующей формулой:

$$f_{NegativeBinomial}(x \mid r, p) = \frac{\Gamma(r+x)}{\Gamma(r)\Gamma(x+1)} p^r (1-p)^x$$

где:

- x — значение случайной величины
- r — количество успешных испытаний
- p — вероятность успеха



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityNegativeBinomial	Рассчитывает плотность вероятности отрицательного биномиального распределения
MathCumulativeDistributionNegativeBinomial	Рассчитывает значение функции отрицательного биномиального распределения вероятностей
MathQuantileNegativeBinomial	Рассчитывает значение обратной функции отрицательного биномиального распределения для заданной вероятности
MathRandomNegativeBinomial	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по отрицательному биномиальному закону
MathMomentsNegativeBinomial	Рассчитывает теоретические численные значения первых 4 моментов отрицательного биномиального распределения

© 2000-2025, MetaQuotes Ltd.

Пример:

```
#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;         // количество испытаний
input double p_par=0.75;       // вероятность успеха для каждого испытания
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // количество значений в выборке
   int ncells=19;          // количество интервалов в гистограмме
   double x[];             // центры интервалов гистограммы
   double y[];             // количество значений из выборки, попавших в интервал
   double data[];          // выборка случайных значений
   double max,min;         // максимальное и минимальное значения в выборке
//--- получим выборку из отрицательного биномиального распределения
   MathRandomNegativeBinomial(n_par,p_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityNegativeBinomial(x2,n_par,p_par,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   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
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNegativeBinomial

Рассчитывает значение функции вероятности (probability mass function) отрицательного биномиального распределения с параметрами r и p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // значение случайной величины (целочисленное)
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) отрицательного биномиального распределения с параметрами r и p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // значение случайной величины (целочисленное)
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) отрицательного биномиального распределения с параметрами r и p для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dnbinom() в R.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // массив со значениями случайной величины
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает значение функции вероятности (probability mass function) отрицательного биномиального распределения с параметрами r и p для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // массив со значениями случайной величины
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

© 2000-2025, MetaQuotes Ltd.

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

r

[in] Количество успешных испытаний

p

[in] Вероятность успеха.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNegativeBinomial

Рассчитывает функцию распределения для отрицательного биномиального закона с параметрами r и p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // значение случайной величины (целочисленное)
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает функцию распределения для отрицательного биномиального закона с параметрами r и p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // значение случайной величины (целочисленное)
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает функцию распределения для отрицательного биномиального закона с параметрами r и p для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pweibull() в R.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // массив со значениями случайной величины
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает функцию распределения для отрицательного биномиального закона с параметрами r и p для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // массив со значениями случайной величины
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] Массив со значениями случайной величины.

r

[in] Количество успешных испытаний.

p

[in] Вероятность успеха.

tail

[in] Флаг расчета, если true, то то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета, логарифма значения, если log_mode=true, то рассчитывается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNegativeBinomial

Рассчитывает обратное значение функции распределения для отрицательного биномиального закона с параметрами r и p для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // значение вероятности появления случайной величины
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения для отрицательного биномиального закона с параметрами r и p для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // значение вероятности появления случайной величины
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения для отрицательного биномиального закона с параметрами r и p для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qnbinom() в R.

```
double  MathQuantileNegativeBinomial(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает обратное значение функции распределения для отрицательного биномиального закона с параметрами r и p для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileNegativeBinomial(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

© 2000-2025, MetaQuotes Ltd.

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

r

[in] Количество успешных испытаний.

p

[in] Вероятность успеха.

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomNegativeBinomial

Генерирует псевдослучайную величину, распределенную по закону отрицательного биномиального распределения с параметрами r и p. В случае ошибки возвращает NaN.

```
double  MathRandomNegativeBinomial(
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону отрицательного биномиального распределения с параметрами r и p. В случае ошибки возвращает false. Аналог rweibull() в R.

```
bool  MathRandomNegativeBinomial(
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

r

[in] Количество успешных испытаний.

p

[in] Вероятность успеха.

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNegativeBinomial

Рассчитывает теоретические численные значения первых 4 моментов отрицательного биномиального распределения с параметрами r и p.

```
double  MathMomentsNegativeBinomial(
   const double  r,              // количество успешных испытаний
   const double  p,              // вероятность успеха
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

r

[in] Количество успешных испытаний.

p

[in] Вероятность успеха.

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

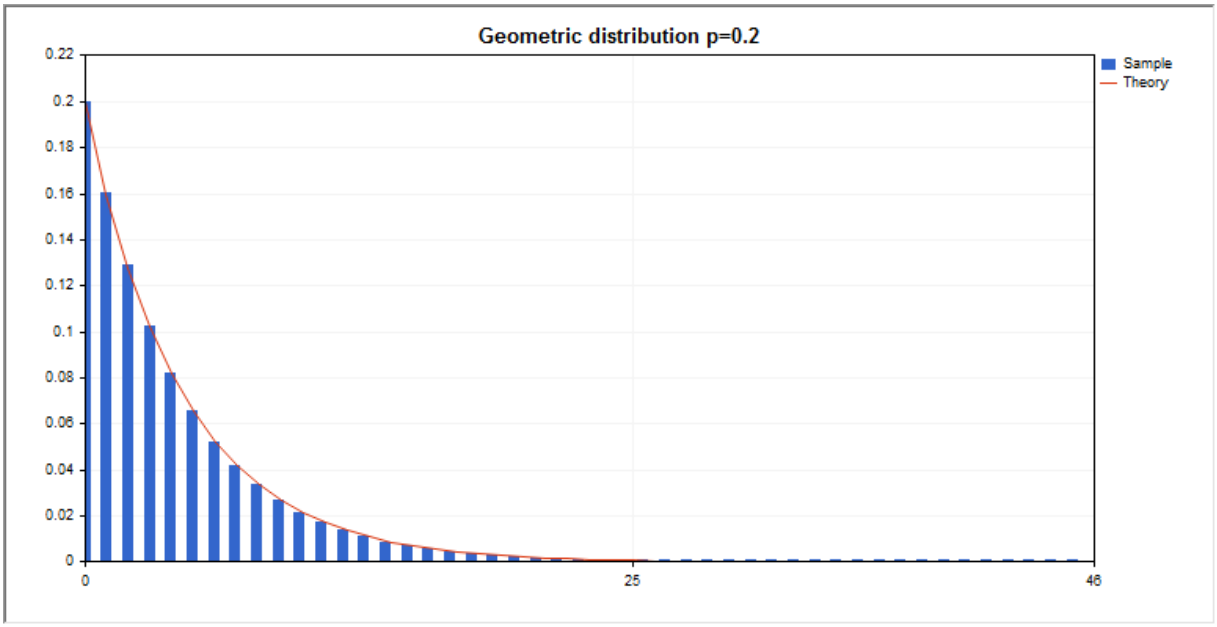
Геометрическое распределение

В данном разделе представлены функции для работы с геометрическим распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по геометрическому закону. Геометрическое распределение описывается следующей формулой:

$$f_{Geometric}(x|p) = p(1-p)^x$$

где:

- x — значение случайной величины (целочисленное)
- p — вероятность появления события в одном опыте



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityGeometric	Рассчитывает плотность вероятности геометрического распределения
MathCumulativeDistributionGeometric	Рассчитывает значение функции геометрического распределения вероятностей
MathQuantileGeometric	Рассчитывает значение обратной функции геометрического распределения для заданной вероятности
MathRandomGeometric	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по геометрическому закону
MathMomentsGeometric	Рассчитывает теоретические численные значения первых 4 моментов геометрического распределения

© 2000-2025, MetaQuotes Ltd.

Пример:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Geometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double p_par=0.2;      // вероятность появления события в одном опыте
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // количество значений в выборке
   int ncells=47;       // количество интервалов в гистограмме
   double x[];          // центры интервалов гистограммы
   double y[];          // количество значений из выборки, попавших в интервал
   double data[];       // выборка случайных значений
   double max,min;      // максимальное и минимальное значения в выборке
//--- получим выборку из геометрического  распределения
   MathRandomGeometric(p_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(0,ncells,1,x2);
   MathProbabilityDensityGeometric(x2,p_par,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Geometric distribution p=%G",p_par));
   graphic.BackgroundMainSize(16);
//--- отключим автомасштабирование оси 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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   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                       |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityGeometric

Рассчитывает значение функции вероятности (probability mass function) геометрического распределения с параметром p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // значение случайной величины (целочисленное)
   const double  p,              // параметр распределения (вероятность появления событ
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) геометрического распределения с параметром p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // значение случайной величины (целочисленное)
   const double  p,              // параметр распределения (вероятность появления событ
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) геометрического распределения с параметром p для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dgeom() в R.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // массив со значениями случайной величины
   const double  p,              // параметр распределения (вероятность появления собы
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает значение функции вероятности (probability mass function) геометрического распределения с параметром p для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // массив со значениями случайной величины
   const double  p,              // параметр распределения (вероятность появления собы
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

p

[in] Параметр распределения (вероятность появления события в одном опыте).

© 2000-2025, MetaQuotes Ltd.

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionGeometric

Рассчитывает функцию распределения для геометрического закона с параметром p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // значение случайной величины (целочисленное)
   const double  p,              // параметр распределения (вероятность появления событ
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает функцию распределения для геометрического закона с параметром p для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // значение случайной величины (целочисленное)
   const double  p,              // параметр распределения (вероятность появления событ
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает функцию распределения для геометрического закона с параметром p для массива случайных величин x[]. В случае ошибки возвращает false. Аналог pgeom() в R.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // массив со значениями случайной величины
   const double  p,              // параметр распределения (вероятность появления собы
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции вероятности
   );
```

Рассчитывает функцию распределения для геометрического закона с параметром p для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // массив со значениями случайной величины
   const double  p,              // параметр распределения (вероятность появления собы
   double&       result[]        // массив для значений функции вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

p

[in] Параметр распределения (вероятность появления события в одном опыте).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Флаг расчета, если tail=true, то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета логарифма значения, если log_mode=true, то рассчитывается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции вероятности.

© 2000-2025, MetaQuotes Ltd.

MathQuantileGeometric

Рассчитывает обратное значение функции распределения для геометрического закона с параметром p для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // значение вероятности появления случайной величины
   const double  p,              // параметр распределения (вероятность появления собы
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения для геометрического закона с параметром p для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // значение вероятности появления случайной величины
   const double  p,              // параметр распределения (вероятность появления собы
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения для геометрического закона с параметром p для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qgeom() в R.

```
double  MathQuantileGeometric(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  p,              // параметр распределения (вероятность появления собы
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает обратное значение функции распределения для геометрического закона с параметром p для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileGeometric(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  p,              // параметр распределения (вероятность появления собы
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

p

© 2000-2025, MetaQuotes Ltd.

[in] Параметр распределения (вероятность появления события в одном опыте).

tail

[in] Флаг расчета, если false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomGeometric

Генерирует псевдослучайную величину, распределенную по закону геометрического распределения с параметром p. В случае ошибки возвращает NaN.

```
double  MathRandomGeometric(
   const double  p,              // параметр распределения (вероятность появления собы
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону геометрического распределения с параметром p. В случае ошибки возвращает false. Аналог rgeom() в R.

```
bool  MathRandomGeometric(
   const double  p,              // параметр распределения (вероятность появления собы
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

p

[in] Параметр распределения (вероятность появления события в одном опыте).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsGeometric

Рассчитывает теоретические численные значения первых 4 моментов геометрического распределения с параметром p.

```
double  MathMomentsGeometric(
   const double  p,              // параметр распределения (вероятность появления собы
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

p

[in] Параметр распределения (вероятность появления события в одном опыте).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

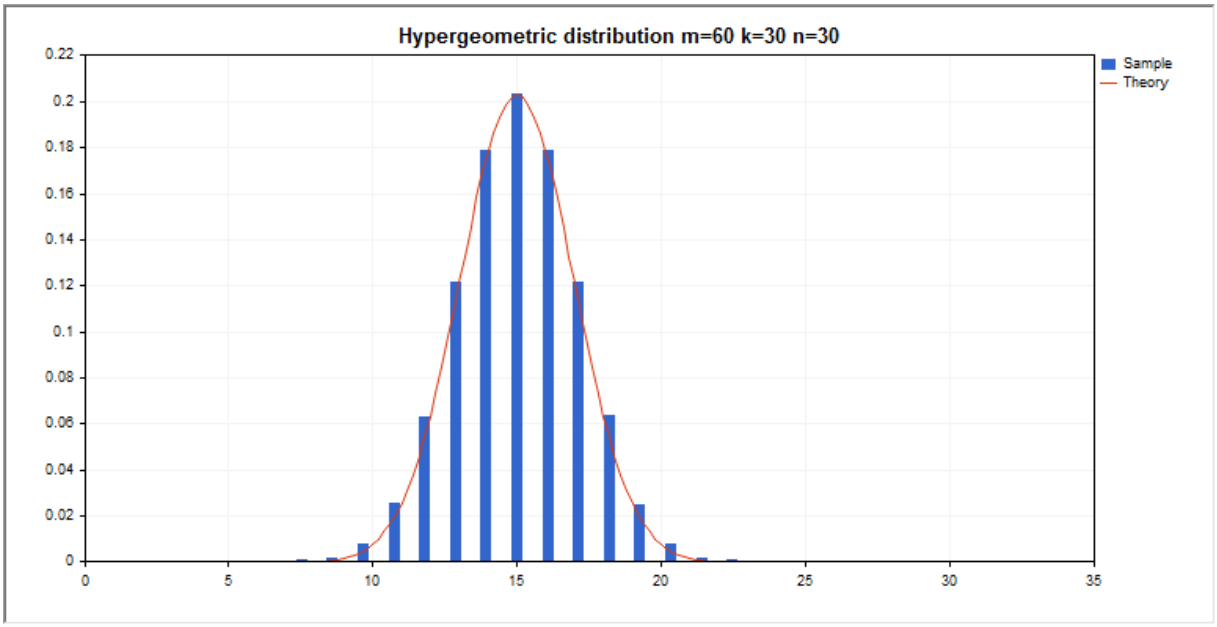
Гипергеометрическое распределение

В данном разделе представлены функции для работы с гипергеогеометрическим распределением. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по гипергеометрическому закону. Гипергеометрическое распределение описывается следующей формулой:

$$f_{Hypergeometric}(x \mid m, k, n) = \frac{\binom{k}{x}\binom{m-k}{n-x}}{\binom{m}{n}}$$

где:

- x — значение случайной величины (целочисленное)
- m — общее количество объектов
- k — количество объектов с желаемой характеристикой
- n — количество взятых объектов



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityHypergeometric	Рассчитывает плотность вероятности гипергеометрического распределения
MathCumulativeDistributionHypergeometric	Рассчитывает значение функции гипергеометрического распределения вероятностей
MathQuantileHypergeometric	Рассчитывает значение обратной функции гипергеометрического распределения для заданной вероятности

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathRandomHypergeometric	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по гипергеометрическому закону
MathMomentsHypergeometric	Рассчитывает теоретические численные значения первых 4 моментов гипергеометрического распределения

Пример:

```
#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;       // общее количество объектов
input double k_par=30;       // количество объектов с желаемой характеристикой
input double n_par=30;       // количество взятых объектов
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // количество значений в выборке
   int ncells=15;        // количество интервалов в гистограмме
   double x[];           // центры интервалов гистограммы
   double y[];           // количество значений из выборки, попавших в интервал
   double data[];        // выборка случайных значений
   double max,min;       // максимальное и минимальное значения в выборке
//--- получим выборку из гипергеометрического  распределения
   MathRandomHypergeometric(m_par,k_par,n_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityHypergeometric(x2,m_par,k_par,n_par,false,y2);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- масштабируем
   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;
//--- выводим графики
   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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

© 2000-2025, MetaQuotes Ltd.

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityHypergeometric

Рассчитывает значение функции вероятности (probability mass function) гипергеометрического распределения с параметрами m, k и n для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,             // значение случайной величины (целочисленное)
   const double  m,             // общее количество объектов (целочисленное)
   const double  k,             // количество объектов с желаемой характеристикой (цел
   const double  n,             // количество взятых объектов (целочисленное)
   const bool    log_mode,      // расчет логарифма значения, если log_mode=true, то в
   int&          error_code     // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) гипергеометрического распределения с параметрами m, k и n для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,             // значение случайной величины (целочисленное)
   const double  m,             // общее количество объектов (целочисленное)
   const double  k,             // количество объектов с желаемой характеристикой (цел
   const double  n,             // количество взятых объектов (целочисленное)
   int&          error_code     // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) гипергеометрического распределения с параметрами m, k и n для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dhyper() в R.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],           // массив со значениями случайной величины
   const double  m,             // общее количество объектов (целочисленное)
   const double  k,             // количество объектов с желаемой характеристикой (це
   const double  n,             // количество взятых объектов (целочисленное)
   const bool    log_mode,      // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]       // массив для значений функции плотности вероятности
   );
```

Рассчитывает значение функции вероятности (probability mass function) гипергеометрического распределения с параметрами m, k и n для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],           // массив со значениями случайной величины
   const double  m,             // общее количество объектов (целочисленное)
   const double  k,             // количество объектов с желаемой характеристикой (це
   const double  n,             // количество взятых объектов (целочисленное)
   double&       result[]       // массив для значений функции плотности вероятности
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

m

[in] Общее количество объектов (целочисленное).

k

[in] Количество объектов с желаемой характеристикой (целочисленное).

n

[in] Количество взятых объектов (целочисленное).

log_mode

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionHypergeometric

Рассчитывает значение функции распределения для гипергеометрического закона с параметрами m, k и n для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // значение случайной величины (целочисленное)
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (цел
   const double  n,              // количество взятых объектов (целочисленное)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции распределения для гипергеометрического закона с параметрами m, k и n для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // значение случайной величины (целочисленное)
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (цел
   const double  n,              // количество взятых объектов (целочисленное)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции распределения для гипергеометрического закона с параметрами m, k и n для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dhyper() в R.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // массив со значениями случайной величины
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (це
   const double  n,              // количество взятых объектов (целочисленное)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции распределения
   );
```

Рассчитывает значение функции распределения для гипергеометрического закона с параметрами m, k и n для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // массив со значениями случайной величины
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (це
   const double  n,              // количество взятых объектов (целочисленное)
   double&       result[]        // массив для значений функции распределения
   );
```

© 2000-2025, MetaQuotes Ltd.

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

m

[in] Общее количество объектов (целочисленное).

k

[in] Количество объектов с желаемой характеристикой (целочисленное).

n

[in] Количество взятых объектов (целочисленное).

tail

[in] Флаг расчета, если true, то то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета, логарифма значения, если log_mode=true, то рассчитывается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции распределения.

© 2000-2025, MetaQuotes Ltd.

MathQuantileHypergeometric

Рассчитывает обратное значение функции распределения для гипергеометрического закона с параметрами m, k и n для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // значение вероятности появления случайной величины
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (це
   const double  n,              // количество взятых объектов (целочисленное)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения для гипергеометрического закона с параметрами m, k и n для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // значение вероятности появления случайной величины
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (це
   const double  n,              // количество взятых объектов (целочисленное)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения для гипергеометрического закона с параметрами m, k и n для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qhyper() в R.

```
double  MathQuantileHypergeometric(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (це
   const double  n,              // количество взятых объектов (целочисленное)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает обратное значение функции распределения для гипергеометрического закона с параметрами m, k и n для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantileHypergeometric(
   const double& probability[],  // массив со значениями вероятностей случайной величи
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (це
   const double  n,              // количество взятых объектов (целочисленное)
   double&       result[]        // массив со значениями квантилей
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

m

[in] Общее количество объектов (целочисленное).

k

[in] Количество объектов с желаемой характеристикой (целочисленное).

n

[in] Количество взятых объектов (целочисленное).

tail

[in] Флаг расчета, если tail=false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomHypergeometric

Генерирует псевдослучайную величину, распределенную по закону гипергеометрического распределения с параметрами m, n и k. В случае ошибки возвращает NaN.

```
double  MathRandomHypergeometric(
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (це
   const double  n,              // количество взятых объектов (целочисленное)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенные по закону гипергеометрического распределения с параметрами m, n и k. В случае ошибки возвращает false. Аналог rgeom() в R.

```
bool  MathRandomHypergeometric(
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (це
   const double  n,              // количество взятых объектов (целочисленное)
   const int     data_count,     // количество необходимых данных
   double&       result[]        // массив со значениями псевдослучайных величин
   );
```

Параметры

m

[in] Общее количество объектов (целочисленное).

k

[in] Количество объектов с желаемой характеристикой (целочисленное).

n

[in] Количество взятых объектов (целочисленное).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsHypergeometric

Рассчитывает теоретические численные значения первых 4 моментов гипергеометрического распределения с параметрами m, n и k.

```
double  MathMomentsHypergeometric(
   const double  m,              // общее количество объектов (целочисленное)
   const double  k,              // количество объектов с желаемой характеристикой (це
   const double  n,              // количество взятых объектов (целочисленное)
   double&       mean,           // переменная для среднего значения
   double&       variance,       // переменная для дисперсии
   double&       skewness,       // переменная для коэффициента асимметрии
   double&       kurtosis,       // переменная для коэффициента эксцесса
   int&          error_code      // переменная для кода ошибки
   );
```

Параметры

m

[in] Общее количество объектов (целочисленное).

k

[in] Количество объектов с желаемой характеристикой (целочисленное).

n

[in] Количество взятых объектов (целочисленное).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

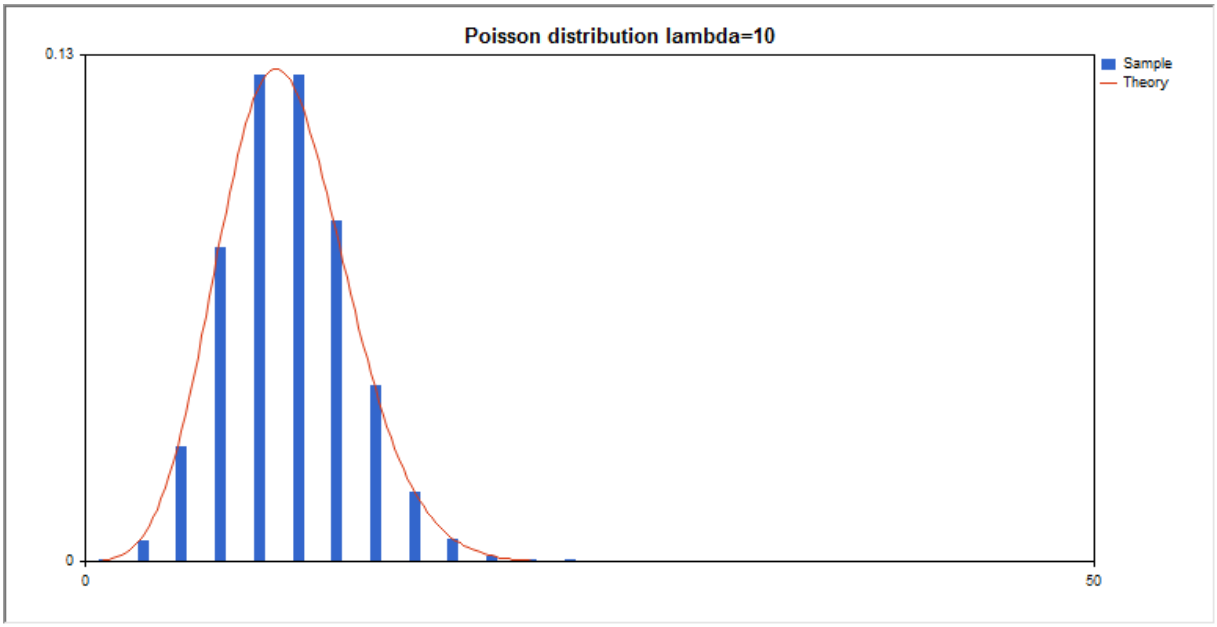
Распределение Пуассона

В данном разделе представлены функции для работы с распределением Пуассона. Они позволяют производить расчет плотности, вероятности, квантилей и генерировать псевдослучайные числа, распределенные по закону Пуассона. Распределение Пуассона описывается следующей формулой:

$$f_{Poisson}(x|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}$$

где:

- x — значение случайной величины
- λ — параметр распределения (mean)



Помимо расчета отдельных случайных величин, реализована возможность работы с их массивами.

Функция	Описание
MathProbabilityDensityPoisson	Рассчитывает плотность вероятности распределения Пуассона
MathCumulativeDistributionPoisson	Рассчитывает значение функции распределения вероятностей Пуассона
MathQuantilePoisson	Рассчитывает значение обратной функции распределения Пуассона для заданной вероятности
MathRandomPoisson	Генерирует псевдослучайную величину/массив псевдослучайных величин, распределенных по закону Пуассона
MathMomentsPoisson	Рассчитывает теоретические численные значения первых 4 моментов распределения Пуассона

Пример:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Poisson.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double lambda_par=10;      // параметр распределения (mean)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- отключим показ ценового графика
   ChartSetInteger(0,CHART_SHOW,false);
//--- инициализируем генератор случайных чисел
   MathSrand(GetTickCount());
//--- сгенерируем выборку случайной величины
   long chart=0;
   string name="GraphicNormal";
   int n=100000;        // количество значений в выборке
   int ncells=13;       // количество интервалов в гистограмме
   double x[];          // центры интервалов гистограммы
   double y[];          // количество значений из выборки, попавших в интервал
   double data[];       // выборка случайных значений
   double max,min;      // максимальное и минимальное значения в выборке
//--- получим выборку из распределения Пуассона
   MathRandomPoisson(lambda_par,n,data);
//--- рассчитаем данные для построения гистограммы
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- получим границы последовательности и шаг для построения теоретической кривой
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- получим теоретически рассчитанные данные на интервале [min,max]
   double x2[];
   double y2[];
   MathSequence(0,int(MathCeil(max)),1,x2);
   MathProbabilityDensityPoisson(x2,lambda_par,false,y2);
//--- масштабируем
   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;
//--- выводим графики
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("Poisson distribution lambda=%G",lambda_par));
   graphic.BackgroundMainSize(16);
//--- отключим автомасштабирование оси 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);
//--- а теперь построим теоретическую кривую плотности распределения
   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);
//--- зададим центры интервалов
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- заполним частоты попадания в интервал
   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)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- вычислим абсолютный размах последовательности, чтобы получить точность нормализа
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- нормализуем макс. и мин. значения с заданной точностью
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- шаг генерации последовательности также зададим от заданной точности
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityPoisson

Рассчитывает значение функции вероятности (probability mass function) распределения Пуассона с параметром lambda для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // значение случайной величины (целочисленное)
   const double  lambda,         // параметр распределения (mean)
   const bool    log_mode,       // расчет логарифма значения, если log_mode=true, то в
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) распределения Пуассона с параметром lambda для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // значение случайной величины (целочисленное)
   const double  lambda,         // параметр распределения (mean)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции вероятности (probability mass function) распределения Пуассона с параметром lambda для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dhyper() в R.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // массив со значениями случайной величины
   const double  lambda,         // параметр распределения (mean)
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Рассчитывает значение функции вероятности (probability mass function) распределения Пуассона с параметром lambda для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // массив со значениями случайной величины
   const double  lambda,         // параметр распределения (mean)
   double&       result[]        // массив для значений функции плотности вероятности
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

lambda

[in] Параметр распределения (mean).

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] Флаг расчета логарифма значения. Если log_mode=true, то возвращается натуральный логарифм плотности вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции плотности вероятности.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionPoisson

Рассчитывает значение функции распределения Пуассона с параметром lambda для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // значение случайной величины (целочисленное)
   const double  lambda,         // параметр распределения (mean)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятно
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=true
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции распределения Пуассона с параметром lambda для случайной величины x. В случае ошибки возвращает NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // значение случайной величины (целочисленное)
   const double  lambda,         // параметр распределения (mean)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает значение функции распределения Пуассона с параметром lambda для массива случайных величин x[]. В случае ошибки возвращает false. Аналог dhyper() в R.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // массив со значениями случайной величины
   const double  lambda,         // параметр распределения (mean)
   const double  tail,           // флаг расчета, если true, то рассчитывается вероятн
   const bool    log_mode,       // флаг расчета логарифма значения, если log_mode=tru
   double&       result[]        // массив для значений функции распределения
   );
```

Рассчитывает значение функции распределения Пуассона с параметром lambda для массива случайных величин x[]. В случае ошибки возвращает false.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // массив со значениями случайной величины
   const double  lambda,         // параметр распределения (mean)
   double&       result[]        // массив для значений функции распределения
   );
```

Параметры

x

[in] Значение случайной величины.

x[]

[in] Массив со значениями случайной величины.

lambda

[in] Параметр распределения (mean).

© 2000-2025, MetaQuotes Ltd.

tail

[in] Флаг расчета, если true, то то рассчитывается вероятность того, что случайная величина не превысит x.

log_mode

[in] Флаг расчета, логарифма значения, если log_mode=true, то рассчитывается натуральный логарифм вероятности.

error_code

[out] Переменная для записи кода ошибки.

result[]

[out] Массив для значений функции распределения.

© 2000-2025, MetaQuotes Ltd.

MathQuantilePoisson

Рассчитывает обратное значение функции распределения Пуассона с параметром lambda для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // значение вероятности появления случайной величины
   const double  lambda,         // параметр распределения (mean)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения Пуассона с параметром lambda для вероятности *probability*. В случае ошибки возвращает NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // значение вероятности появления случайной величины
   const double  lambda,         // параметр распределения (mean)
   int&          error_code      // переменная для записи кода ошибки
   );
```

Рассчитывает обратное значение функции распределения Пуассона с параметром lambda для массива значений вероятности *probability[]*. В случае ошибки возвращает false. Аналог qhyper() в R.

```
double  MathQuantilePoisson(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  lambda,         // параметр распределения (mean)
   const bool    tail,           // флаг расчета, если false, то расчет ведется для ве
   const bool    log_mode,       // флаг расчета, если log_mode=true, то расчет ведетс
   double&       result[]        // массив со значениями квантилей
   );
```

Рассчитывает обратное значение функции распределения Пуассона с параметром lambda для массива значений вероятности *probability[]*. В случае ошибки возвращает false.

```
bool  MathQuantilePoisson(
   const double& probability[],  // массив со значениями вероятностей случайной велич
   const double  lambda,         // параметр распределения (mean)
   double&       result[]        // массив со значениями квантилей
   );
```

Параметры

probability

[in] Значение вероятности случайной величины.

probability[]

[in] Массив со значениями вероятностей случайной величины.

lambda

© 2000-2025, MetaQuotes Ltd.

[in] Параметр распределения (mean).

tail

[in] Флаг расчета, если tail=false, то расчет ведется для вероятности 1.0-probability.

log_mode

[in] Флаг расчета, если log_mode=true, то расчет ведется для вероятности Exp(probability).

error_code

[out] Переменная для получения кода ошибки.

result[]

[out] Массив со значениями квантилей.

© 2000-2025, MetaQuotes Ltd.

MathRandomPoisson

Генерирует псевдослучайную величину, распределенную по закону распределения Пуассона с параметром lambda. В случае ошибки возвращает NaN.

```
double  MathRandomPoisson(
   const double  lambda,          // параметр распределения (mean)
   int&          error_code       // переменная для записи кода ошибки
   );
```

Генерирует псевдослучайные величины, распределенныепо закону распределения Пуассона с параметром lambda. В случае ошибки возвращает false. Аналог rgeom() в R.

```
bool  MathRandomPoisson(
   const double  lambda,          // параметр распределения (mean)
   const int     data_count,      // количество необходимых данных
   double&       result[]         // массив со значениями псевдослучайных величин
   );
```

Параметры

lambda

[in] Параметр распределения (mean).

error_code

[out] Переменная для записи кода ошибки.

data_count

[out] Количество необходимых данных.

result[]

[out] Массив для получения значений псевдослучайных величин.

© 2000-2025, MetaQuotes Ltd.

MathMomentsPoisson

Рассчитывает теоретические численные значения первых 4 моментов распределения Пуассона с параметром lambda.

```
double  MathMomentsPoisson(
   const double  lambda,          // параметр распределения (mean)
   double&       mean,            // переменная для среднего значения
   double&       variance,        // переменная для дисперсии
   double&       skewness,        // переменная для коэффициента асимметрии
   double&       kurtosis,        // переменная для коэффициента эксцесса
   int&          error_code       // переменная для кода ошибки
   );
```

Параметры

lambda

[in] Параметр распределения (mean).

mean

[out] Переменная для получения среднего значения.

variance

[out] Переменная для получения дисперсии.

skewness

[out] Переменная для получения коэффициента асимметрии.

kurtosis

[out] Переменная для получения коэффициента эксцесса.

error_code

[out] Переменная для получения кода ошибки.

Возвращаемое значение

Возвращает true, если расчет моментов произведен успешно, иначе false.

© 2000-2025, MetaQuotes Ltd.

Вспомогательные функции

Группа функций, выполняющих основные математические операции: расчет гамма-функции, бета-функции, факториала, экспоненты, логарифмов с различными основаниями, квадратного корня и т.п.

Предоставлены возможности обработки как отдельных числовых значений (вещественных и целочисленных), так и их массивов (с записью результатов в отдельный или в исходный массив).

Функция	Описание
MathRandomNonZero	Возвращает случайное число в диапазоне от 0.0 до 1.0 с плавающей запятой.
MathMoments	Рассчитывает первые 4 момента элементов массива: среднее, дисперсию, коэффициент асимметрии, коэффициент эксцесса.
MathPowInt	Возводит число в указанную целочисленную степень.
MathFactorial	Рассчитывает факториал заданного целого числа.
MathTrunc	Рассчитывает целую часть заданного числа или элементов массива.
MathRound	Округляет число или массив чисел до заданного количества дробных разрядов.
MathArctan2	Рассчитывает угол, тангенс которого равен отношению двух указанных чисел в диапазоне [-pi, pi].
MathGamma	Вычисляет значение гамма-функции.
MathGammaLog	Вычисляет логарифм гамма-функции.
MathBeta	Вычисляет значение бета-функции.
MathBetaLog	Вычисляет значение логарифма бета-функции.
MathBetaIncomplete	Рассчитывает значение неполной бета-функции.
MathGammaIncomplete	Рассчитывает значение неполной гамма-функции.
MathBinomialCoefficient	Рассчитывает биномиальный коэффицент.
MathBinomialCoefficientLog	Рассчитывает логарифм биномиального коэффицента.
MathHypergeometric2F2	Рассчитывает значение гипергеометрической функции.

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathSequence	Формирует последовательность на основе значений: первый элемент, последний элемент, шаг последовательности.
MathSequenceByCount	Формирует последовательность на основе значений: первый элемент, последний элемент, количество элементов последовательности.
MathReplicate	Формирует повторяющуюся последовательность значений.
MathReverse	Формирует массив значений с обратным порядком элементов.
MathIdentical	Сравнивает два массива значений и возвращает true, если совпадают все элементы.
MathUnique	Формирует массив только с неповторяющимися значениями.
MathQuickSortAscending	Функция для восходящей сортировки.
MathQuickSortDescending	Функция для нисходящей сортировки.
MathQuickSort	Функция для сортировки.
MathOrder	Формирует массив с перестановкой в соответствии с порядком элементов массива после сортировки.
MathBitwiseNot	Рассчитывает результат бинарной операции NOT для элементов массива.
MathBitwiseAnd	Рассчитывает результат бинарной операции AND для элементов массивов.
MathBitwiseOr	Рассчитывает результат бинарной операции OR для элементов массивов.
MathBitwiseXor	Рассчитывает результат бинарной операции XOR для элементов массивов.
MathBitwiseShiftL	Рассчитывает результат бинарной операции SHL для элементов массива.
MathBitwiseShiftR	Рассчитывает результат бинарной операции SHR для элементов массива.
MathCumulativeSum	Формирует массив с накопленной суммой.
MathCumulativeProduct	Формирует массив с накопленным произведением.

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathCumulativeMin	Формирует массив с накопленными минимальными значениями.
MathCumulativeMax	Формирует массив с накопленными максимальными значениями.
MathSin	Рассчитывает значение функции sin(x) для элементов массива.
MathCos	Рассчитывает значение функции cos(x) для элементов массива.
MathTan	Рассчитывает значение функции tan(x) для элементов массива.
MathArcsin	Рассчитывает значение функции arcsin(x) для элементов массива.
MathArccos	Рассчитывает значение функции arccos(x) для элементов массива.
MathArctan	Рассчитывает значение функции arctan(x) для элементов массива.
MathSinPi	Рассчитывает значение функции sin(pi*x) для элементов массива.
MathCosPi	Рассчитывает значение функции cos(pi*x) для элементов массива.
MathTanPi	Рассчитывает значение функции tan(pi*x) для элементов массива.
MathAbs	Рассчитывает абсолютное значение элементов массива.
MathCeil	Возвращает ближайшее сверху целое числовое значение для элементов массива.
MathFloor	Возвращает ближайшее снизу целое числовое значение для элементов массива.
MathSqrt	Рассчитывает квадратный корень для элементов массива.
MathExp	Рассчитывает значение функции exp(x) для элементов массива.
MathPow	Рассчитывает значение функции pow(x, power) для элементов массива.
MathLog	Рассчитывает значение функции log(x) для элементов массива.
MathLog2	Рассчитывает значение логарифма по основанию 2 для элементов массива.

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathLog10	Рассчитывает значение логарифма по основанию 10 для элементов массива.
MathDifference	Формирует массив с разностями элементов y[i]=x[i+lag]-x[i].
MathSample	Производит случайную выборку элементов массива.
MathTukeySummary	Рассчитывает пятичисловую сводку Тьюки для элементов массива.
MathRange	Рассчитывает минимальные и максимальные значения элементов массива.
MathMin	Возвращает минимальное значение среди всех элементов массива.
MathMax	Возвращает максимальное значение среди всех элементов массива.
MathSum	Возвращает сумму элементов массива.
MathProduct	Возвращает произведение элементов массива.
MathStandardDeviation	Рассчитывает стандартное отклонение элементов массива.
MathAverageDeviation	Рассчитывает среднее отклонение элементов массива.
MathMedian	Рассчитывает медианное значение элементов массива.
MathMean	Рассчитывает среднее значение элементов массива.
MathVariance	Рассчитывает дисперсию элементов массива.
MathSkewness	Рассчитывает коэффициент асимметрии элементов массива.
MathKurtosis	Рассчитывает коэффициент эксцесса элементов массива.
MathLog1p	Рассчитывает значение функции log(1+x) для элементов массива.
MathExpm1	Рассчитывает значение функции exp(x)-1 для элементов массива.
MathSinh	Рассчитывает значение функции sinh(x) для элементов массива.

© 2000-2025, MetaQuotes Ltd.

Функция	Описание
MathCosh	Рассчитывает значение функции cosh(x) для элементов массива.
MathTanh	Рассчитывает значение функции tanh(x) для элементов массива.
MathArcsinh	Рассчитывает значение функции arcsinh(x) для элементов массива.
MathArccosh	Рассчитывает значение функции arccosh(x) для элементов массива.
MathArctanh	Рассчитывает значение функции arctanh(x) для элементов массива.
MathSignif	Округляет значение до указанного количества знаков в мантиссе.
MathRank	Рассчитывает ранги элементов массива.
MathCorrelationPearson	Рассчитывает коэффициент корреляции Пирсона.
MathCorrelationSpearman	Рассчитывает коэффициент корреляции Спирмена.
MathCorrelationKendall	Рассчитывает коэффициент корреляции Кендалла.
MathQuantile	Рассчитывает выборочные квантили, соответствующие указанным вероятностям.
MathProbabilityDensityEmpirical	Вычисляет эмпирическую плотность вероятности для случайных значений.
MathCumulativeDistributionEmpirical	Вычисляет эмпирическое кумулятивное распределение для случайных значений.

© 2000-2025, MetaQuotes Ltd.

MathRandomNonZero

Возвращает случайное число в диапазоне от 0.0 до 1.0 с плавающей запятой.

```
double  MathRandomNonZero()
```

Возвращаемое значение

Случайное число в диапазоне от 0.0 до 1.0 с плавающей запятой.

© 2000-2025, MetaQuotes Ltd.

MathMoments

Рассчитывает первые 4 момента элементов массива: среднее, дисперсию, коэффициент асимметрии, коэффициент эксцесса.

```
bool  MathMoments(
   const double&  array[],            // массив значений
   double&        mean,               // переменная для среднего
   double&        variance,           // переменная для дисперсии
   double&        skewness,           // переменная для коэффициента асимметрии
   double&        kurtosis,           // переменная для коэффициента эксцесса
   const int      start=0,            // начальный индекс
   const int      count=WHOLE_ARRAY   // количество элементов
   )
```

Параметры

array[]

[in] Массив значений.

mean

[out] Переменная для среднего значения (1 момент).

variance

[out] Переменная для дисперсии (2 момент).

skewness

[out] Переменная для коэффициента асимметрии (3 момент).

kurtosis

[out] Переменная для коэффициента эксцесса (4 момент).

start=0

[in] Начальный индекс для расчета.

count=WHOLE_ARRAY

[in] Количество элементов для расчета.

Возвращаемое значение

Возвращает true, если моменты успешно рассчитаны, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathPowInt

Возводит число в указанную целочисленную степень.

```
double  MathPowInt(
   const double  x,       // значение числа
   const int     power    // степень возведения
   )
```

Параметры

x

[in] Число двойной точности с плавающей запятой, возводимое в степень.

power

[in] Целое число, задающее степень.

Возвращаемое значение

Число x, возведенное в указанную степень.

© 2000-2025, MetaQuotes Ltd.

MathFactorial

Рассчитывает факториал заданного целого числа.

```
double  MathFactorial(
   const int  n   // значение числа
   )
```

Параметры

n

[in] Целое число, факториал котого необходимо вычислить.

Возвращаемое значение

Факториал числа.

© 2000-2025, MetaQuotes Ltd.

MathTrunc

Рассчитывает целую часть заданного числа или элементов массива.

Версия для работы с числом двойной точности с плавающей запятой:

```
double  MathTrunc(
   const double   x          // значение числа
   )
```

Возвращаемое значение

Целая часть заданного числа.

Версия для работы с массивом чисел двойной точности с плавающей запятой. Результаты записываются в новый массив:

```
bool  MathTrunc(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Версия для работы с массивом чисел двойной точности с плавающей запятой. Результаты записываются в этот же массив:

```
bool  MathTrunc(
   double&        array[]    // массив значений
   )
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Параметры

x

[in] Число двойной точности с плавающей запятой, целую часть которого нужно получить.

array[]

[in] Массив чисел двойной точности с плавающей запятой, целую часть которых нужно получить.

array[]

[out] Массив выходных значений.

result[]

[out] Массив выходных значений.

© 2000-2025, MetaQuotes Ltd.

MathRound

Округляет число двойной точности с плавающей запятой или массив таких чисел до заданного количества дробных разрядов.

Версия для округления числа двойной точности с плавающей запятой до заданного количества дробных разрядов:

```
double  MathRound(
   const double   x,          // значение числа
   const int      digits      // количество знаков после запятой
   )
```

Возвращаемое значение

Число, ближайшее к параметру x, количество цифр дробной части которого равно digits.

Версия для округления массива чисел двойной точности с плавающей запятой до заданного количества дробных разрядов. Результаты записываются в новый массив.

```
bool  MathRound(
   const double&  array[],    // массив значений
   int            digits,     // количество знаков после запятой
   double&        result[]    // массив результатов
   )
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Версия для округления массива чисел двойной точности с плавающей запятой до заданного количества дробных разрядов. Результаты записываются в этот же массив.

```
bool  MathRound(
   double&        array[],    // массив значений
   int            digits      // количество знаков после запятой
   )
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Параметры

x

[in] Округляемое число двойной точности с плавающей запятой.

digits

[in] Количество цифр дробной части в возвращаемом значении.

array[]

[in] Массив округляемых чисел двойной точности с плавающей запятой.

array[]

© 2000-2025, MetaQuotes Ltd.

[out] Массив выходных значений.

result[]

[out] Массив выходных значений.

© 2000-2025, MetaQuotes Ltd.

MathArctan2

Возвращает арктангенс от частного двух аргументов (x, y).

Версия для работы с отношением двух указанных чисел (x, y):

```
double  MathArctan2(
   const double       y,          // координата y
   const double       x           // координата x
   )
```

Возвращаемое значение

Угол, θ, измеренный в радианах, такой, что -π≤θ≤π, и tan (θ) = y или x, где (x, y) — это точка в декартовой системе координат.

Версия для работы с отношением пары элементов из массивов x и y:

```
bool  MathArctan2(
   const double&      x[],        // массив значений x
   const double&      y[],        // массив значений y
   double&            result[]    // массив результатов
   )
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Параметры

y

[in] Координата y точки.

x

[in] Координата x точки.

x[]

[in] Массив координат x точек.

y[]

[in] Массив координат y точек.

result[]

[out] Массив для записи результатов

Примечания

Обратите внимание на следующее.

- Для (x, y) в квадранте 1 возвращаемое значение будет: 0 < θ < π/2.
- Для (x, y) в квадранте 2 возвращаемое значение будет: π/2 < θ≤π.
- Для (x, y) в квадранте 3 возвращаемое значение будет: -π < θ <-π/2.
- Для (x, y) в квадранте 4 возвращаемое значение будет: -π/2 < θ < 0.

© 2000-2025, MetaQuotes Ltd.

Для точек за пределами указанных квадрантов возвращаемое значение указано ниже.

- Если y равно 0 и x не является отрицательным, θ = 0.
- Если y равно 0 и x является отрицательным, θ = π.
- Если y — положительное число, а x равно 0, θ = π/2.
- Если y является отрицательным и x равно 0, θ = -π/2.
- Если y равен 0 и x равен 0, то θ = -π/2.

Если значение параметра x или y равно NaN либо если значения параметров x и y равны значению PositiveInfinity или NegativeInfinity, метод возвращает значение NaN.

© 2000-2025, MetaQuotes Ltd.

MathGamma

Вычисляет значение гамма-функции для вещественного аргумента x.

```
double  MathGamma(
   const double  x       // аргумент функции
   )
```

Параметры

x

[in] Вещественный аргумент функции.

Возвращаемое значение

Значение гамма-функции.

© 2000-2025, MetaQuotes Ltd.

MathGammaLog

Вычисляет значение логарифма гамма-функции для вещественного аргумента x.

```
double  MathGammaLog(
   const double  x      // аргумент функции
   )
```

Параметры

x

[in] Вещественный аргумент функции.

Возвращаемое значение

Значение логарифма функции.

© 2000-2025, MetaQuotes Ltd.

MathBeta

Вычисляет значение бета-функции для вещественных аргументов a и b.

```
double  MathBeta(
   const double  a,       // первый аргумент функции
   const double  b        // второй аргумент функции
   )
```

Параметры

a

[in] Аргумент функции a.

b

[in] Аргумент функции b.

Возвращаемое значение

Значение функции.

© 2000-2025, MetaQuotes Ltd.

MathBetaLog

Вычисляет значение логарифма бета-функции для вещественных аргументов a и b.

```
double  MathBetaLog(
   const double  a,       // первый аргумент функции
   const double  b        // второй аргумент функции
   )
```

Параметры

a

[in] Аргумент функции a.

b

[in] Аргумент функции b.

Возвращаемое значение

Значение логарифма функции.

© 2000-2025, MetaQuotes Ltd.

MathBetaIncomplete

Рассчитывает значение неполной бета-функции.

```
double  MathBetaIncomplete(
   const double  x,       // аргумент функции
   const double  p,       // первый параметр функции
   const double  q        // второй параметр функции
   )
```

Параметры

x

[in] Аргумент функции.

p

[in] Первый параметр бета-функции, должен быть >0.0.

q

[in] Второй параметр бета-функции, должен быть >0.0.

Возвращаемое значение

Значение функции.

© 2000-2025, MetaQuotes Ltd.

MathGammaIncomplete

Вычисляет значение функции неполной гамма-функции.

```
double  MathGammaIncomplete(
   double  x,           // аргумент функции
   double  alpha        // параметр функции
   )
```

Параметры

x

[in] Аргумент функции.

alpha

[in] Параметр неполной гамма-функции.

Возвращаемое значение

Значение функции.

© 2000-2025, MetaQuotes Ltd.

MathBinomialCoefficient

Рассчитывает биномиальный коэффицент: C(n,k)=n!/(k!*(n-k)!).

```
long  MathBinomialCoefficient(
   const int  n,       // общее количество элементов
   const int  k        // количество элементов в сочетании
   )
```

Параметры

n

[in] Количество элементов.

k

[in] Количество элементов для каждого сочетания.

Возвращаемое значение

Число комбинаций из N по K.

© 2000-2025, MetaQuotes Ltd.

MathBinomialCoefficientLog

Рассчитывает логарифм биномиального коэффицента: Log(C(n,k))=Log(n!/(k!*(n-k)!))

Версия для целочисленных аргументов:

```
double  MathBinomialCoefficientLog(
   const int      n,       // общее количество элементов
   const int      k        // количество элементов в сочетании
   )
```

Версия для вещественных аргументов:

```
double  MathBinomialCoefficientLog(
   const double   n,       // общее количество элементов
   const double   k        // количество элементов в сочетании
   )
```

Параметры

n

[in] Количество элементов.

k

[in] Количество элементов для каждого сочетания.

Возвращаемое значение

Логарифм от C(n,k).

© 2000-2025, MetaQuotes Ltd.

MathHypergeometric2F2

Расчитывает значение функции Hypergeometric_2F2 (a, b, c, d, z), используя метод Тейлора.

```
double  MathHypergeometric2F2(
   const double  a,      // первый параметр функции
   const double  b,      // второй параметр функции
   const double  c,      // третий параметр функции
   const double  d,      // четвертый параметр функции
   const double  z       // пятый параметр функции
   )
```

Параметры

a

[in] Первый параметр функции.

b

[in] Второй параметр функции.

c

[in] Третий параметр функции.

d

[in] Четвертый параметр функции.

z

[in] Пятый параметр функции.

Возвращаемое значение

Значение функции.

© 2000-2025, MetaQuotes Ltd.

MathSequence

Формирует последовательность значений на основе данных значений: первый элемент, последний элемент, шаг последовательности.

Версия для работы с вещественными значениями:

```
bool  MathSequence(
   const double  from,        // начальное значение
   const double  to,          // конечное значение
   const double  step,        // шаг
   double&       result[]     // массив результатов
   )
```

Версия для работы с целочисленными значениями:

```
bool  MathSequence(
   const int     from,        // начальное значение
   const int     to,          // конечное значение
   const int     step,        // шаг
   int&          result[]     // массив результатов
   )
```

Параметры

from

[in] Первое значение последовательности

to

[in] Последнее значение последовательности

step

[in] Шаг последовательности.

result[]

[out] Массив для записи последовательности.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathSequenceByCount

Формирует последовательность значений на основе данных значений: первый элемент, последний элемент, количество элементов последовательности.

Версия для работы с вещественными значениями:

```
bool  MathSequenceByCount(
   const double  from,        // начальное значение
   const double  to,          // конечное значение
   const int     count,       // количество
   double&       result[]     // массив результатов
   )
```

Версия для работы с целочисленными значениями:

```
bool  MathSequenceByCount(
   const int     from,        // начальное значение
   const int     to,          // конечное значение
   const int     count,       // количество
   int&          result[]     // массив результатов
   )
```

Параметры

from

[in] Первое значение последовательности.

to

[in] Последнее значение последовательности.

count

[in] Количество элементов последовательности.

result[]

[out] Массив для записи последовательности.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathReplicate

Формирует повторяющуюся последовательность значений.

Версия для работы с вещественными значениями:

```
bool  MathReplicate(
   const double&  array[],    // массив значений
   const int      count,      // количество повторов
   double&        result[]    // массив результатов
   )
```

Версия для работы с целочисленными значениями:

```
bool  MathReplicate(
   const int&     array[],    // массив значений
   const int      count,      // количество повторов
   int&           result[]    // массив результатов
   )
```

Параметры

array[]

[in] Массив для формирования последовательности.

count

[in] Количество повторений массива в последовательности.

result[]

[out] Массив для записи последовательности.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathReverse

Формирует массив значений с обратным порядком элементов.

Версия для работы с вещественными значениями с сохранением результатов в новый массив:

```
bool  MathReverse(
   const double&  array[],      // массив значений
   double&        result[]      // массив результатов
   )
```

Версия для работы с целочисленными значениями с сохранением результатов в новый массив:

```
bool  MathReverse(
   const int&     array[],      // массив значений
   int&           result[]      // массив результатов
   )
```

Версия для работы с вещественными значениями с сохранением результатов в тот же массив.

```
bool  MathReverse(
   double&        array[]       // массив значений
   )
```

Версия для работы с целочисленными значениями с сохранением результатов в тот же массив.

```
bool  MathReverse(
   int&           array[]       // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

array[]

[out] Выходной массив с обратным порядком значений.

result[]

[out] Выходной массив с обратным порядком значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathIdentical

Сравнивает два массива значений и возвращает true, если совпадают все элементы.

Версия для работы с массивами вещественных значений:

```
bool  MathIdentical(
   const double&  array1[],      // первый массив значений
   const double&  array2[]       // второй массив значений
   )
```

Версия для работы с массивами целочисленных значений:

```
bool  MathIdentical(
   const int&     array1[],      // первый массив значений
   const int&     array2[]       // второй массив значений
   )
```

Параметры

array1[]

[in] Первый массив для сравнения.

array2[]

[in] Второй массив для сравнения.

Возвращаемое значение

Возвращает true, если массивы равны, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathUnique

Формирует массив только с неповторяющимися значениями.

Версия для работы с вещественными значениями:

```
bool  MathUnique(
   const double&   array[],   // массив значений
   double&         result[]   // массив результатов
   )
```

Версия для работы с целочисленными значениями:

```
bool  MathUnique(
   const int&     array[],      // массив значений
   int&           result[]      // массив результатов
   )
```

Параметры

array[]

[in] Исходный массив.

result[]

[out] Массив для записи уникальных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathQuickSortAscending

Функция для одновременной восходящей сортировки массивов array[] и indices[], использующая алгоритм QuickSort.

```
void  MathQuickSortAscending(
   double&  array[],     // массив значений
   int&     indices[],   // массив индексов
   int      first,       // начальное значение
   int      last         // конечное значение
   )
```

Параметры

array[]

[in][out] Массив для сортировки.

indices[]

[in][out] Массив для сохранения индексов исходного массива.

first

[in] Индекс элемента, с которого нужно начать сортировку.

last

[in] Индекс элемента, на котором нужно закончить сортировку.

© 2000-2025, MetaQuotes Ltd.

MathQuickSortDescending

Функция для одновременной нисходящей сортировки массивов array[] и indices[], использущая алгоритм QuickSort.

```
void  MathQuickSortDescending(
   double&  array[],      // массив значений
   int&     indices[],    // массив индексов
   int      first,        // начальное значение
   int      last          // конечное значение
   )
```

Параметры

array[]

[in][out] Массив для сортировки.

indices[]

[in][out] Массив для сохранения индексов исходного массива.

first

[in] Индекс элемента, с которого нужно начать сортировку.

last

[in] Индекс элемента, на котором нужно закончить сортировку.

© 2000-2025, MetaQuotes Ltd.

MathQuickSort

Функция для одновременной сортировки массивов array[] и indices[], использущая алгоритм QuickSort.

```
void  MathQuickSort(
   double&  array[],      // массив значений
   int&     indices[],    // массив индексов
   int      first,        // начальное значение
   int      last,         // конечное значение
   int      mode          // направление
   )
```

Параметры

array[]

[in][out] Массив для сортировки.

indices[]

[in][out] Массив для сохранения индексов исходного массива.

first

[in] Индекс элемента, с которого нужно начать сортировку.

last

[in] Индекс элемента, на котором нужно закончить сортировку.

mode

[in] Направление сортировки (>0 восходящая, иначе нисходящая).

© 2000-2025, MetaQuotes Ltd.

MathOrder

Формирует целочисленный массив с перестановкой в соответствии с порядком элементов массива после сортировки.

Версия для работы с массивом вещественных значений:

```
bool  MathOrder(
   const double&  array[],   // массив значений
   int&           result[]   // массив результатов
   )
```

Версия для работы с массивом целочисленных значений:

```
bool  MathOrder(
   const int&     array[],   // массив значений
   int&           result[]   // массив результатов
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив для записи отсортированных индексов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseNot

Рассчитывает результат бинарной операции NOT для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathBitwiseNot(
   const int&  array[],   // массив значений
   int&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathBitwiseNot(
   int&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

array[]

[out] Массив выходных значений.

result[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseAnd

Рассчитывает результат бинарной операции AND для заданных массивов.

```
bool  MathBitwiseAnd(
   const int&  array1[],  // первый массив значений
   const int&  array2[],  // второй массив значений
   int&        result[]   // массив результатов
   )
```

Параметры

array1[]

[in] Первый массив значений.

array2[]

[in] Второй массив значений.

result[]

[out] Массив для записи результатов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseOr

Рассчитывает результат бинарной операции OR для заданных массивов.

```
bool  MathBitwiseOr(
   const int&  array1[],  // первый массив значений
   const int&  array2[],  // второй массив значений
   int&        result[]   // массив результатов
   )
```

Параметры

array1[]

[in] Первый массив значений.

array2[]

[in] Второй массив значений.

result[]

[out] Массив для записи результатов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseXor

Рассчитывает результат бинарной операции XOR для заданных массивов.

```
bool  MathBitwiseXor(
   const int&  array1[],  // первый массив значений
   const int&  array2[],  // второй массив значений
   int&        result[]   // массив результатов
   )
```

Параметры

array1[]

[in] Первый массив значений.

array2[]

[in] Второй массив значений.

result[]

[out] Массив для записи результатов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseShiftL

Рассчитывает результат бинарной операции SHL (побитовый сдвиг влево) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathBitwiseShiftL(
   const int&  array[],    // массив значений
   const int   n,          // значение сдвига
   int&        result[]    // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathBitwiseShiftL(
   int&        array[],    // массив значений
   const int   n           // значение сдвига
   )
```

Параметры

array[]

[in] Массив значений.

n

[in] Число битов для сдвига.

array[]

[out] Массив выходных значений.

result[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseShiftR

Рассчитывает результат бинарной операции SHR (побитовый сдвиг вправо) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathBitwiseShiftR(
   const int&  array[],    // массив значений
   const int   n,          // значение сдвига
   int&        result[]    // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathBitwiseShiftR(
   int&        array[],    // массив значений
   const int   n           // значение сдвига
   )
```

Параметры

array[]

[in] Массив значений.

n

[in] Число битов для сдвига.

array[]

[out] Массив выходных значений.

result[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeSum

Формирует массив с накопленной суммой.

Версия с записью результатов в новый массив:

```
bool  MathCumulativeSum(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathCumulativeSum(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

array[]

[out] Массив выходных значений.

result[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeProduct

Формирует массив с накопленным произведением.

Версия с записью результатов в новый массив:

```
bool  MathCumulativeProduct(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathCumulativeProduct(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeMin

Формирует массив с накопленными минимальными значениями.

Версия с записью результатов в новый массив:

```
bool  MathCumulativeMin(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathCumulativeMin(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeMax

Формирует массив с накопленными максимальными значениями.

Версия с записью результатов в новый массив:

```
bool  MathCumulativeMax(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathCumulativeMax(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathSin

Рассчитывает значение функции sin(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathSin(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathSin(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCos

Рассчитывает значение функции cos(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathCos(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathCos(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathTan

Рассчитывает значение функции tan(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathTan(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathTan(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathArcsin

Рассчитывает значение функции arcsin(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathArcsin(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathArcsin(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathArccos

Рассчитывает значение функции arccos(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathArccos(
   const double&   array[],    // массив значений
   double&         result[]    // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathArccos(
   double&         array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathArctan

Рассчитывает значение функции arctan(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathArctan(
   const double&   array[],   // массив значений
   double&         result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathArctan(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathSinPi

Рассчитывает значение функции sin(pi*x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathSinPi(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathSinPi(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCosPi

Рассчитывает значение функции cos(pi*x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathCosPi(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathCosPi(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathTanPi

Рассчитывает значение функции tan(pi*x) для элементов массива.

Версия с записью результата в новый массив:

```
bool  MathTanPi(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результата в исходный массив:

```
bool  MathTanPi(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathAbs

Рассчитывает абсолютное значение элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathAbs(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathAbs(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCeil

Возвращает ближайшее сверху целое числовое значение для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathCeil(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathCeil(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathFloor

Возвращает ближайшее снизу целое числовое значение для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathFloor(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathFloor(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathSqrt

Рассчитывает квадратный корень для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathSqrt(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathSqrt(
   double&         array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathExp

Рассчитывает значение функции exp(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathExp(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathExp(
   double&       array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathPow

Рассчитывает значение функции pow(x, power) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathPow(
   const double&  array[],    // массив значений
   const double   power,      // степень
   double&        result[]    // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathPow(
   double&        array[],    // массив значений
   const double   power       // степень
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathLog

Рассчитывает значение функции log(x) для элементов массива.

Версия для расчета натурального логарифма с записью результатов в новый массив.

```
bool  MathLog(
   const double&  array[],      // массив значений
   double&        result[]      // массив результатов
   )
```

Версия для расчета натурального логарифма с записью результатов в исходный массив.

```
bool  MathLog(
   double&        array[]       // массив значений
   )
```

Версия для расчета логарифма по заданному основанию с записью результатов в новый массив.

```
bool  MathLog(
   const double&   array[],     // массив значений
   const double    base,        // основание логарифма
   double&         result[]     // массив результатов
   )
```

Версия для расчета логарифма по заданному основанию с записью результатов в исходный массив.

```
bool  MathLog(
   double&         array[],     // массив значений
   const double    base         // основание логарифма
   )
```

Параметры

array[]

[in] Массив значений.

base

[in] Основание логарифма.

array[]

[out] Массив выходных значений.

result[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathLog2

Рассчитывает значение логарифма по основанию 2 для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathLog2(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathLog2(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathLog10

Рассчитывает значение логарифма по основанию 10 для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathLog10(
   const double&   array[],   // массив значений
   double&         result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathLog10(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathLog1p

Рассчитывает значение функции log(1+x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathLog1p(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathLog1p(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathDifference

Формирует массив с разностями элементов y[i]=x[i+lag]-x[i].

Версия для однократного формирования массива вещественных значений:

```
bool  MathDifference(
   const double  &array[],     // массив значений
   const int     lag,          // запаздывание
   double        &result[]     // массив результатов
   )
```

Версия для однократного формирования массива целочисленных значений:

```
bool  MathDifference(
   const int     &array[],     // массив значений
   const int     lag,          // запаздывание
   int           &result[]     // массив результатов
   )
```

Версия для многократного формирования массива вещественных значений (количество итераций задается во входных параметрах):

```
bool  MathDifference(
   const double  &array[],     // массив значений
   const int     lag,          // запаздывание
   const int     differences,  // количество итераций
   double        &result[]     // массив результатов
   )
```

Версия для многократного формирования массива целочисленных значений (количество итераций задается во входных параметрах):

```
bool  MathDifference(
   const int&    array[],      // массив значений
   const int     lag,          // запаздывание
   const int     differences,  // количество итераций
   int&          result[]      // массив результатов
   )
```

Параметры

array[]

[in] Массив значений.

lag

[in] Параметр запаздывания.

differences

[in] Количество итераций.

result[]

© 2000-2025, MetaQuotes Ltd.

[out] Массив для записи результатов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathSample

Производит случайную выборку элементов массива.

Версия для работы с массивом вещественных значений:

```
bool  MathSample(
   const double&  array[],          // массив значений
   const int      count,            // количество
   double&        result[]          // массив результатов
   )
```

Версия для работы с массивом целочисленных значений:

```
bool  MathSample(
   const int&     array[],          // массив значений
   const int      count,            // количество
   int&           result[]          // массив результатов
   )
```

Версия для работы с массивом вещественных значений. Присутствует возможность получить выборку с возвратом:

```
bool  MathSample(
   const double&  array[],          // массив значений
   const int      count,            // количество
   const bool     replace,          // флаг
   double&        result[],         // массив результатов
   )
```

Версия для работы с массивом целочисленных значений. Присутствует возможность получить выборку с возвратом:

```
bool  MathSample(
   const int&     array[],          // массив значений
   const int      count,            // количество
   const bool     replace,          // флаг
   int&           result[]          // массив результатов
   )
```

Версия для работы с массивом вещественных значений, для которых определены вероятности попадания в выборку.

```
bool  MathSample(
   const double&  array[],          // массив значений
   double&        probabilities[],  // массив вероятностей
   const int      count,            // количество
   double&        result[]          // массив результатов
   )
```

Версия для работы с массивом целочисленных значений, для которых определены вероятности попадания в выборку.

© 2000-2025, MetaQuotes Ltd.

```
bool  MathSample(
   const int&     array[],           // массив значений
   double&        probabilities[],   // массив вероятностей
   const int      count,             // количество
   int&           result[]           // массив результатов
   )
```

Версия для работы с массивом вещественных значений, для которых определены вероятности попадания в выборку. Есть возможность получения выборки с возвратом:

```
bool  MathSample(
   const double&  array[],           // массив значений
   double&        probabilities[],   // массив вероятностей
   const int      count,             // количество
   const bool     replace,           // флаг
   double&        result[]           // массив результатов
   )
```

Версия для работы с массивом целочисленных значений, для которых определены вероятности попадания в выборку. Есть возможность получения выборки с возвратом:

```
bool  MathSample(
   const int&     array[],           // массив значений
   double&        probabilities[],   // массив вероятностей
   const int      count,             // количество
   const bool     replace,           // флаг
   int&           result[]           // массив результатов
   )
```

Параметры

array[]

[in] Массив целочисленных значений.

probabilities[]

[in] Массив вероятностей, с которыми производится выборка элементов.

count

[in] Количество элементов.

replace

[in] Параметр, позволяющий осуществлять отбор с возвратом.

result[]

[out] Массив для записи результатов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

© 2000-2025, MetaQuotes Ltd.

Аргумент replace=true позволяет осуществлять случайный отбор элементов с их возвратом в исходную совокупность.

© 2000-2025, MetaQuotes Ltd.

MathTukeySummary

Рассчитывает пятичисловую сводку Тьюки (минимум, нижний квартиль, среднее, верхний квартиль, максимум) для элементов массива.

```
bool  MathTukeySummary(
   const double&  array[],       // массив значений
   const bool     removeNAN,     // флаг
   double&        minimum,       // минимальное значение
   double&        lower_hinge,   // нижний квартиль
   double&        median,        // среднее значение
   double&        upper_hinge,   // верхний квартиль
   double&        maximum        // максимальное значение
   )
```

Параметры

array[]

[in] Массив вещественных значений.

removeNAN

[in] Флаг, указывающий, нужно ли удалять нечисловые значения.

minimum

[out] Переменная для записи минимального значения.

lower_hinge

[out] Переменная для записи нижнего квартиля.

median

[out] Переменная для записи среднего значения.

upper_hinge

[out] Переменная для записи верхнего квартиля.

maximum

[out] Переменная для записи максимального значения.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathRange

Рассчитывает минимальные и максимальные значения элементов массива.

```
bool  MathRange(
   const double&  array[],  // массив значений
   double&        min,      // минимальное значение
   double&        max       // максимальное значение
   )
```

Параметры

array[]

[in] Массив значений.

min

[out] Переменная для записи минимального значения.

max

[out] Переменная для записи максимального значения.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathMin

Возвращает минимальное значение среди всех элементов массива.

```
double  MathMin(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Минимальное значение.

© 2000-2025, MetaQuotes Ltd.

MathMax

Возвращает максимальное значение среди всех элементов массива.

```
double  MathMax(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Максимальное значение.

© 2000-2025, MetaQuotes Ltd.

MathSum

Возвращает сумму элементов массива.

```
double  MathSum(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Сумма элементов.

© 2000-2025, MetaQuotes Ltd.

MathProduct

Возвращает произведение элементов массива.

```
double  MathProduct(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Произведение элементов.

© 2000-2025, MetaQuotes Ltd.

MathStandardDeviation

Рассчитывает стандартное отклонение элементов массива.

```
double  MathStandardDeviation(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Стандартное отклонение.

© 2000-2025, MetaQuotes Ltd.

MathAverageDeviation

Функция рассчитывает среднее отклонение элементов массива.

```
double  MathAverageDeviation(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Среднее отклонение элементов массива.

© 2000-2025, MetaQuotes Ltd.

MathMedian

Рассчитывает медианное значение элементов массива.

```
double  MathMedian(
   double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Медианное значение.

© 2000-2025, MetaQuotes Ltd.

MathMean

Рассчитывает среднее значение элементов массива.

```
double  MathMean(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Среднее значение.

© 2000-2025, MetaQuotes Ltd.

MathVariance

Функция рассчитывает дисперсию (второй момент) элементов массива.

```
double  MathVariance(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Значение дисперсии.

© 2000-2025, MetaQuotes Ltd.

MathSkewness

Функция рассчитывает коэффициент асимметрии (третий момент) элементов массива.

```
double  MathSkewness(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Коэффицент асимметрии.

© 2000-2025, MetaQuotes Ltd.

MathKurtosis

Функция рассчитывает коэффициент эксцесса (четвертый момент) элементов массива.

```
double  MathKurtosis(
   const double&  array[]   // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

Возвращаемое значение

Коэффицент эксцесса.

© 2000-2025, MetaQuotes Ltd.

MathExpm1

Рассчитывает значение функции exp(x)-1 для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathExpm1(
   const double&   array[],   // массив значений
   double&         result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathExpm1(
   double&         array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathSinh

Рассчитывает значение функции sinh(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathSinh(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathSinh(
   double&       array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCosh

Рассчитывает значение функции cosh(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathCosh(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathCosh(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathTanh

Рассчитывает значение функции tanh(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathTanh(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathTanh(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathArcsinh

Рассчитывает значение функции arcsinh(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathArcsinh(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathArcsinh(
   double&         array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathArccosh

Рассчитывает значение функции arccosh(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathArccosh(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathArccosh(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathArctanh

Рассчитывает значение функции arctanh(x) для элементов массива.

Версия с записью результатов в новый массив:

```
bool  MathArctanh(
   const double&  array[],   // массив значений
   double&        result[]   // массив результатов
   )
```

Версия с записью результатов в исходный массив:

```
bool  MathArctanh(
   double&        array[]    // массив значений
   )
```

Параметры

array[]

[in] Массив значений.

result[]

[out] Массив выходных значений.

array[]

[out] Массив выходных значений.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathSignif

Округляет значение до указанного количества знаков в мантиссе.

Версия для работы с вещественным значением:

```
double  MathSignif(
   const double   x,          // значение
   const int      digits      // количество знаков
   )
```

Возвращаемое значение

Округленное значение.

Версия для работы с массивом вещественных значений с записью результатов в отдельный массив:

```
bool  MathSignif(
   const double&  array[],    // массив значений
   int            digits,     // количество знаков
   double         result[]    // массив результатов
   )
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Версия для работы с массивом вещественных значений с записью результатов в исходный массив:

```
bool  MathSignif(
   double&        array[],    // массив значений
   int            digits      // количество знаков
   )
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Параметры

x

[in] Вещественное значение для округления.

digits

[in] Количество знаков.

array[]

[in] Массив вещественных значений.

array[]

[out] Массив выходных значений.

result[]

[out] Массив выходных значений.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

MathRank

Рассчитывает ранги элементов массива.

Версия для работы с массивом вещественных значений:

```
bool  MathRank(
   const double&  array[],  // массив значений
   double&        rank[]    // массив рангов
   )
```

Версия для работы с массивом целочисленных значений:

```
bool  MathRank(
   const int&     array[],  // массив значений
   double&        rank[]    // массив рангов
   )
```

Параметры

array[]

[in] Массив значений.

rank[]

[out] Массив для записи рангов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCorrelationPearson

Рассчитывает коэффициент корреляции Пирсона.

Версия для работы с массивами вещественных значений:

```
bool  MathCorrelationPearson(
   const double&  array1[],  // первый массив значений
   const double&  array2[],  // второй массив значений
   double&        r          // коэффициент корреляции
   )
```

Версия для работы с массивами целочисленных значений:

```
bool  MathCorrelationPearson(
   const int&     array1[],  // первый массив значений
   const int&     array2[],  // второй массив значений
   double&        r          // коэффициент корреляции
   )
```

Параметры

array1[]

[in] Первый массив значений.

array2[]

[in] Второй массив значений.

r

[out] Переменная для записи коэффициента корреляции.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCorrelationSpearman

Рассчитывает коэффициент корреляции Спирмена.

Версия для работы с массивами вещественных значений:

```
bool  MathCorrelationSpearman(
   const double&  array1[],  // первый массив значений
   const double&  array2[],  // второй массив значений
   double&        r          // коэффициент корреляции
   )
```

Версия для работы с массивами целочисленных значений:

```
bool  MathCorrelationSpearman(
   const int&     array1[],   // первый массив значений
   const int&     array2[],   // второй массив значений
   double&        r           // коэффициент корреляции
   )
```

Параметры

array1[]

[in] Первый массив значений.

array2[]

[in] Второй массив значений.

r

[out] Переменная для записи коэффициента корреляции.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCorrelationKendall

Рассчитывает коэффициент корреляции Кендалла.

Версия для работы с массивами вещественных значений:

```
bool  MathCorrelationKendall(
   const double&  array1[],  // первый массив значений
   const double&  array2[],  // второй массив значений
   double&        tau        // коэффициент корреляции
   )
```

Версия для работы с массивами целочисленных значений:

```
bool  MathCorrelationKendall(
   const int&     array1[],  // первый массив значений
   const int&     array2[],  // второй массив значений
   double&        tau        // коэффициент корреляции
   )
```

Параметры

array1[]

[in] Первый массив значений.

array2[]

[in] Второй массив значений.

tau

[out] Переменная для записи коэффициента корреляции.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathQuantile

Рассчитывает выборочные квантили, соответствующие указанным вероятностям: Q[i] (p) = (1 - gamma)*x[j] + gamma*x[j+1]

```
bool  MathQuantile(
   const double&  array[],     // массив значений
   const double&  probs[],     // массив вероятностей
   double&        quantile[]   // массив для записи квантилей
   )
```

Параметры

array[]

[in] Массив значений.

probs[]

[in] Массив вероятностей.

quantile[]

[out] Массив для записи квантилей.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityEmpirical

Функция вычисляет эмпирическую плотность вероятности (pdf) для случайных значений из массива.

```
bool  MathProbabilityDensityEmpirical(
   const double&  array[],   // массив случайных значений
   const int      count,     // количество пар
   double&        x[],       // массив значений x
   double&        pdf[]      // массив значений pdf
   )
```

Параметры

array[]

[in] Массив случайных значений.

count

[in] Количество пар (x, pdf(x)).

x[]

[out] Массив для записи значений x.

pdf[]

[out] Массив для записи значений pdf(x).

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionEmpirical

Функция вычисляет эмпирическое кумулятивное распределение(cdf) для случайных значений из массива.

```
bool  MathCumulativeDistributionEmpirical(
   const double&  array[],   // массив случайных значений
   const int      count,     // количество пар
   double&        x[],       // массив значений x
   double&        cdf[]      // массив значений cdf
   )
```

Параметры

array[]

[in] Массив случайных значений.

count

[in] Количество пар (x, cdf(x)).

x[]

[out] Массив для записи значений x.

cdf[]

[out] Массив для записи значений cdf(x).

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Fuzzy — библиотека для работы с нечеткой логикой

Нечеткая логика — это обобщение традиционной аристотелевой логики на случай, когда истинность рассматривается как лингвистическая переменная. Как и в классической логике, для нечеткой логики определены свои нечеткие логические операции над нечеткими множествами. Для нечетких множеств существуют все те же операции, что и для обычных множеств, только их вычисление на порядок сложнее. Отметим также, что композиция нечетких множеств — есть нечеткое множество.

Основными особенностями нечеткой логики, отличающими ее от классической, являются максимальная приближенность к отражению реальности и высокий уровень субъективности, вследствие чего могут возникнуть значительные погрешности в результатах вычислений с ее использованием.

Нечеткая модель (или система) — математическая модель, в основе вычисления которой лежит нечеткая логика. К построению таких моделей прибегают в случае, когда предмет исследования имеет очень слабую формализацию, и его точное математическое описание слишком сложное или неизвестно. Качество выходных значений этих моделей (погрешность модели) напрямую зависит только от эксперта, который составлял и настраивал модель. Для минимизации ошибки лучшим вариантом будет составление максимально полной и исчерпывающей модели и последующая ее настройка средствами машинного обучения на большой обучающей выборке.

Ход построения модели можно разделить на три основных этапа:

1. Определение входных и выходных параметров модели.
2. Построение базы знаний.
3. Выбор одного из методов нечеткого логического вывода (Мамдани или Сугено).

От первого этапа непосредственно зависят два других, и именно он определяет будущее функционирование модели. База знаний или, как по-другому ее называют, база правил — это совокупность нечетких правил вида: "если, то", определяющих взаимосвязь между входами и выходами исследуемого объекта. Количество правил в системе не ограниченно и также определяется экспертом. Обобщенный формат нечетких правил такой:

Если условие (посылка) правила, то заключение правила.

Условие правила характеризует текущее состояние объекта, а заключение — то, как это условие повлияет на объект. Общий вид условий и заключений нельзя выделить, так как они определяются нечетким логическим выводом.

Каждое правило в системе имеет вес — данный параметр характеризует значимость правила в модели. Весовые коэффициенты присваиваются правилу в диапазоне [0, 1]. Во многих примерах нечетких моделей, которые можно встретить в литературе, данные веса не указаны, но это не означает, что их нет, в действительности для каждого правила из базы в таком случае вес фиксирован и равен единице. Условия и заключения для каждого правила могут быть двух видов:

1. простое — в нем участвует одна нечеткая переменная;
2. составное — участвуют несколько нечетких переменных.

В зависимости от созданной базы знаний для модели определяется система нечеткого логического вывода. Нечетким логическим выводом называется получение заключения в виде нечеткого множества, соответствующего текущим значениях входов, с использованием нечеткой базы

© 2000-2025, MetaQuotes Ltd.

знаний и нечетких операций. Два основных типа нечеткого логического вывода — Мамдани и Сугено.

© 2000-2025, MetaQuotes Ltd.

Функции принадлежности

Функцией принадлежности *(membership function)* называется функция, которая позволяет вычислить степень принадлежности произвольного элемента универсального множества нечеткому множеству. Следовательно, область значений функции принадлежности должна принадлежать диапазону [0, 1].

В большинстве случаев функция принадлежности монотонна и непрерывна.

Класс функций принадлежности	Описание
CConstantMembershipFunction	Класс для реализации функции принадлежности в виде прямой, параллельной оси координат
CCompositeMembershipFunction	Класс для реализации композиции функций принадлежности
CDifferencTwoSigmoidalMembershipFunction	Класс для реализации функции принадлежности в виде разности между двумя сигмоидными функциями с параметрами A1, A2, C1, C2
CGeneralizedBellShapedMembershipFunction	Класс для реализации обобщенной колоколообразной функции принадлежности с параметрами A, B и C
CNormalCombinationMembershipFunction	Класс для реализации двухсторонней гауссовской функции принадлежности с параметрами B1, B2, Sigma1 и Sigma2
CNormalMembershipFunction	Класс для реализации симметричной гауссовской функции принадлежности с параметрами B и Sigma
CP_ShapedMembershipFunction	Класс для реализации пи-подобной функции принадлежности с параметрами A, B, C и D
CProductTwoSigmoidalMembershipFunction	Класс для реализации функции принадлежности в виде произведения двух сигмоидных функций с параметрами A1, A2, C1, C2
CS_ShapedMembershipFunction	Класс для реализации S-подобной функции принадлежности с параметрами A и B
CTrapezoidMembershipFunction	Класс для реализации трапециевидной функции принадлежности с параметрами X1, X2, X3 и X4
CTriangularMembershipFunction	Класс для реализации треугольной функции принадлежности с параметрами X1, X2, X3
CSigmoidalMembershipFunction	Класс для реализации сигмоидной функции принадлежности с параметрами A и C

© 2000-2025, MetaQuotes Ltd.

Класс функций принадлежности	Описание
CZ_ShapedMembershipFunction	Класс для реализации z-подобной функции принадлежности с параметрами A и B.
IMembershipFunction	Базовый класс для всех классов функций принадлежности.

© 2000-2025, MetaQuotes Ltd.

CConstantMembershipFunction

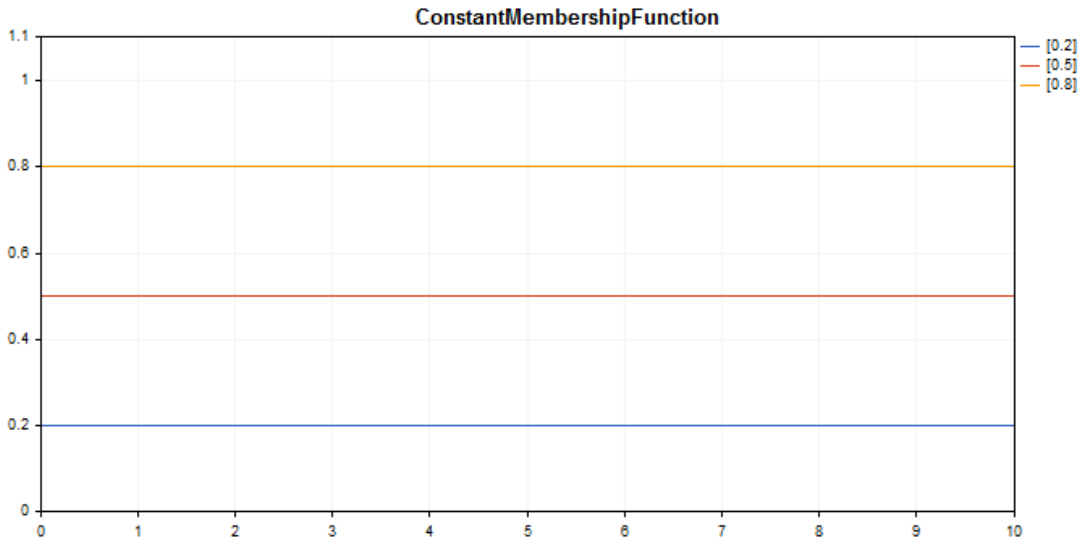
Класс для реализации функции принадлежности в виде прямой, параллельной оси координат.

Описание

Функция описывается уравнением:

y(x)=c

Следовательно, степень принадлежности для данной функции одинакова на всей числовой оси и равняется параметру, указанному в конструкторе.



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CConstantMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CConstantMembershipFunction

Методы класса

Метод класса	Описание
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                                  ConstantMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const double  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

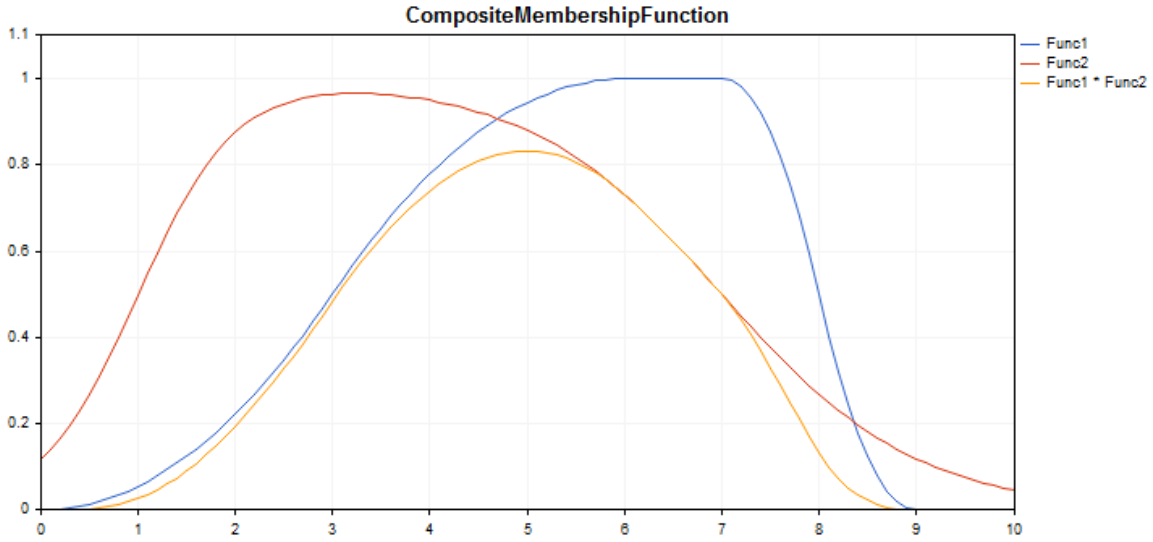
© 2000-2025, MetaQuotes Ltd.

CCompositeMembershipFunction

Класс для реализации композиции функций принадлежности.

Описание

Композиция функций принадлежности — объединение двух или более функций принадлежности с помощью заданного оператора.



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CCompositeMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject
  IMembershipFunction
    CCompositeMembershipFunction

Методы класса

Метод класса	Описание
CompositionType	Устанавливает оператор композиции.
MembershipFunctions	Возвращает список функций принадлежности.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                                  CompositeMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

CompositionType

Устанавливает оператор композиции.

```
void  CompositionType(
   MfCompositionType  value      // тип оператора
   )
```

Параметры

value

[in] Тип оператора композиции.

Примечание

Доступны следующие типы операторов:

- MinMF (минимум функций)
- MaxMF (максимум функций)
- ProdMF (произведение функций)
- SumMF (сумма функций)

MembershipFunctions

Возвращает список функций принадлежности, входящих в композицию.

```
CList*  MembershipFunctions(
   void        // список функций принадлежности
   )
```

Возвращаемое значение

Список функций принадлежности.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Значение функции принадлежности.

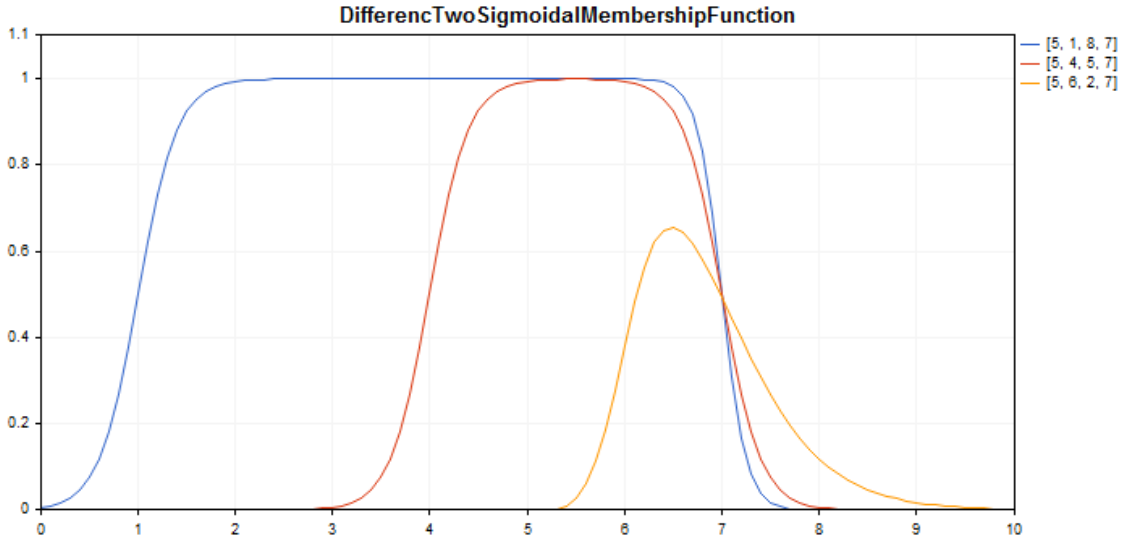
© 2000-2025, MetaQuotes Ltd.

CDifferencTwoSigmoidalMembershipFunction

Класс для реализации функции принадлежности в виде разности между двумя сигмоидными функциями с параметрами A1, A2, C1, C2.

Описание

Функция основана на использовании сигмоидной кривой. С ее помощью можно создавать функции принадлежности со значениями, равными 1, начиная с некоторого значения аргумента. Подобные функции подходят, если необходимо задать лингвистические термы типа "короткий" или "длинный".



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CDifferencTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CDifferencTwoSigmoidalMembershipFunction

Методы класса

Метод класса	Описание
A1	Возвращает и устанавливает коэффициент крутизны первой функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
A2	Возвращает и устанавливает коэффициент крутизны второй функции принадлежности.
C1	Возвращает и устанавливает параметр координаты перегиба первой функции принадлежности.
C2	Возвращает и устанавливает параметр координаты перегиба второй функции принадлежности.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                      DifferencTwoSigmoidalMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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 DifferencTwoSigmoidalMembershipFunction2(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,
```

© 2000-2025, MetaQuotes Ltd.

```
   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 (Метод Get)

Возвращает коэффициент крутизны первой функции принадлежности.

```
double  A1()
```

Возвращаемое значение

Значение коэффициента крутизны.

A1 (Метод Set)

Устанавливает коэффициент крутизны первой функции принадлежности.

```
void  A1(
   const double  a1      // значение коэффициента крутизны
   )
```

Параметры

a1

[in] Значение коэффициента крутизны.

A2 (Метод Get)

Возвращает коэффициент крутизны второй функции принадлежности.

```
double  A2()
```

Возвращаемое значение

Значение коэффициента крутизны.

A2 (Метод Set)

© 2000-2025, MetaQuotes Ltd.

Устанавливает коэффициент крутизны второй функции принадлежности.

```
void  A2(
   const double  a2      // значение коэффициента крутизны
   )
```

Параметры

a2

[in] Значение коэффициента крутизны.

C1 (Метод Get)

Возвращает параметр координаты перегиба первой функции принадлежности.

```
double  C1()
```

Возвращаемое значение

Значение координаты перегиба.

C1 (Метод Set)

Устанавливает параметр координаты перегиба первой функции принадлежности.

```
void  C1(
   const double  c1      // значение координаты перегиба
   )
```

Параметры

c1

[in] Значение координаты перегиба.

C2 (Метод Get)

Возвращает параметр координаты перегиба второй функции принадлежности.

```
double  C2()
```

Возвращаемое значение

Значение координаты перегиба.

C2 (Метод Set)

Устанавливает параметр координаты перегиба второй функции принадлежности.

```
void  C2(
   const double  c2      // значение координаты перегиба
   )
```

© 2000-2025, MetaQuotes Ltd.

Параметры

c2

[in] Значение координаты перегиба.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const double  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

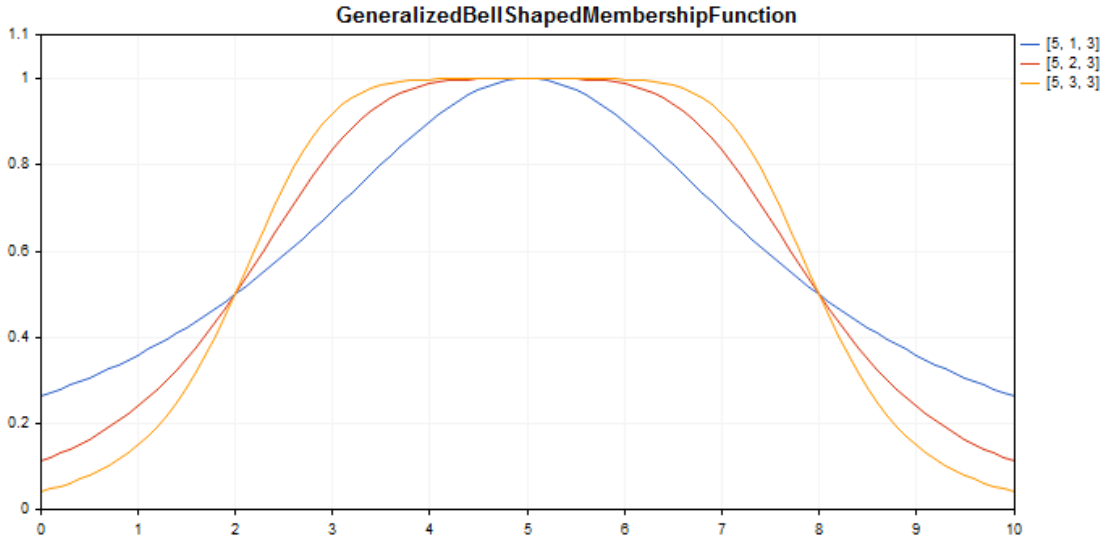
© 2000-2025, MetaQuotes Ltd.

CGeneralizedBellShapedMembershipFunction

Класс для реализации обобщенной колоколообразной функции принадлежности с параметрами A, B и C.

Описание

Обобщенная колоколообразная функция принадлежности по форме похожа на гауссовские. На всей области определения функция является гладкой и принимает ненулевые значения.



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CGeneralizedBellShapedMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CGeneralizedBellShapedMembershipFunction

Методы класса

Метод класса	Описание
A	Возвращает и устанавливает коэффициент концентрации функции принадлежности.
B	Возвращает и устанавливает коэффициент крутизны функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
C	Возвращает и устанавливает координату максимума функции принадлежности.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                      GeneralizedBellShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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);
```

© 2000-2025, MetaQuotes Ltd.

```
   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 (Метод Get)

Возвращает коэффициент концентрации функции принадлежности.

```
double  A()
```

Возвращаемое значение

Значение коэффициента концентрации.

A (Метод Set)

Устанавливает коэффициент концентрации функции принадлежности.

```
void  A(
   const double  a      // значение коэффициента концентрации
   )
```

Параметры

a

[in] Коэффициент концентрации функции принадлежности.

B (Метод Get)

Возвращает коэффициент крутизны функции принадлежности.

```
double  B()
```

Возвращаемое значение

Значение коэффициента крутизны.

B (Метод Set)

Устанавливает коэффициент крутизны функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

```
void  B(
   const double  b      // значение коэффициента крутизны
   )
```

Параметры

b

[in] Коэффициент крутизны функции принадлежности.

C (Метод Get)

Возвращает координату максимума функции принадлежности.

```
double  C()
```

Возвращаемое значение

Координата максимума функции принадлежности.

C (Метод Set)

Устанавливает координату максимума функции принадлежности.

```
void  C(
   const double  c      // значение координаты максимума
   )
```

Параметры

c

[in] Координата максимума функции принадлежности.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

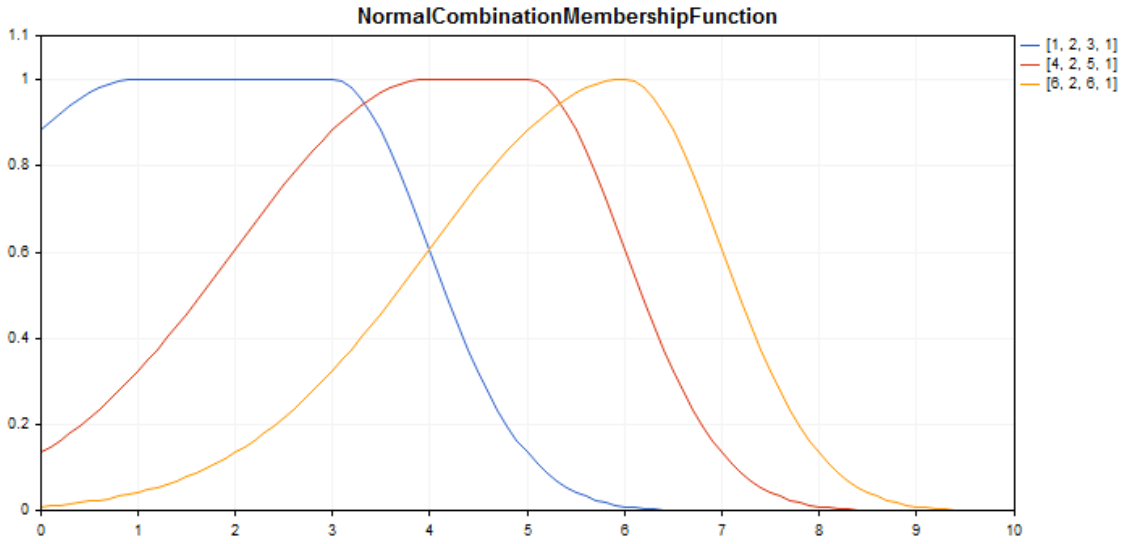
© 2000-2025, MetaQuotes Ltd.

CNormalCombinationMembershipFunction

Класс для реализации двухсторонней гауссовской функции принадлежности с параметрами B1, B2, Sigma1 и Sigma2.

Описание

Двухсторонняя гауссовская функция принадлежности формируется с использованием распределения Гаусса. Она позволяет задавать асимметричные функции принадлежности. На всей области определения функция является гладкой и принимает ненулевые значения.



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CNormalCombinationMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CNormalCombinationMembershipFunction

Методы класса

Метод класса	Описание
B1	Возвращает и устанавливает значение первого центра функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
B2	Возвращает и устанавливает значение второго центра функции принадлежности.
Sigma1	Возвращает и устанавливает первый параметр кривизны функции принадлежности.
Sigma2	Возвращает и устанавливает второй параметр кривизны функции принадлежности.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                          NormalCombinationMembershipFunction.mq5 |
//|                       Copyright 2016, MetaQuotes Software Corp.  |
//|                                             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,
```

© 2000-2025, MetaQuotes Ltd.

```
   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 (Метод Get)

Возвращает значение первого центра функции принадлежности.

```
double  B1()
```

Возвращаемое значение

Значение первого центра функции принадлежности.

B1 (Метод Set)

Устанавливает значение первого центра функции принадлежности.

```
void  B1(
   const double  b1      // значение первого центра
   )
```

Параметры

b

[in] Значение первого центра функции принадлежности.

B2 (Метод Get)

Возвращает значение второго центра функции принадлежности.

```
double  B2()
```

Возвращаемое значение

Значение второго центра функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

B2 (Метод Set)

Устанавливает значение второго центра функции принадлежности.

```
void  B2(
   const double  b2      // значение второго центра
   )
```

Параметры

b2

[in] Значение второго центра функции принадлежности.

Sigma1 (Метод Get)

Возвращает первый параметр кривизны функции принадлежности.

```
double  Sigma1()
```

Возвращаемое значение

Значение первого параметра кривизны функции принадлежности.

Sigma1 (Метод Set)

Устанавливает значение первого параметра кривизны функции принадлежности.

```
void  Sigma1(
   const double  sigma1      // значение первого параметра кривизны
   )
```

Параметры

sigma1

[in] Первый параметр кривизны функции принадлежности.

Sigma2 (Метод Get)

Возвращает второй параметр кривизны функции принадлежности.

```
double  Sigma2()
```

Возвращаемое значение

Значение второго параметра кривизны функции принадлежности.

Sigma2 (Метод Set)

© 2000-2025, MetaQuotes Ltd.

Устанавливает значение второго параметра кривизны функции принадлежности.

```
void  Sigma2(
   const double  sigma2      // значение второго параметра кривизны
   )
```

Параметры

sigma2

[in] Второй параметр кривизны функции принадлежности.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

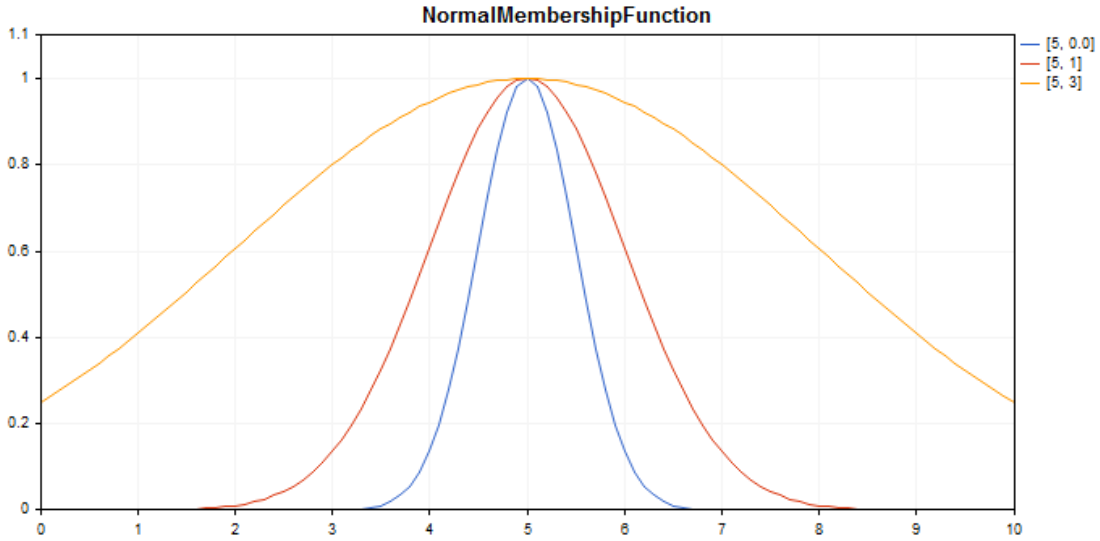
© 2000-2025, MetaQuotes Ltd.

CNormalMembershipFunction

Класс для реализации симметричной гауссовской функции принадлежности с параметрами B и Sigma.

Описание

Симметричная гауссовская функция принадлежности формируется с использованием распределения Гаусса. На всей области определения функция является гладкой и принимает ненулевые значения.



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CNormalMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CNormalMembershipFunction

Методы класса

Метод класса	Описание
B	Возвращает и устанавливает центр функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
Sigma	Возвращает и устанавливает параметр кривизны функции принадлежности.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                                      NormalMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              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);
```

© 2000-2025, MetaQuotes Ltd.

```
   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 (Метод Get)

Возвращает центр функции принадлежности.

```
double  B()
```

Возвращаемое значение

Значение центра функции принадлежности.

B (Метод Set)

Устанавливает значение центра функции принадлежности.

```
void  B(
   const double  b      // значение центра функции
   )
```

Параметры

b

[in] Значение центра функции принадлежности.

Sigma (Метод Get)

Возвращает параметр кривизны функции принадлежности

```
double  Sigma()
```

Возвращаемое значение

Значение параметра кривизны функции принадлежности

Sigma (Метод Set)

Устанавливает значение параметра кривизны функции принадлежности.

```
void  Sigma(
   const double  sigma      // значение параметра кривизны
   )
```

© 2000-2025, MetaQuotes Ltd.

Параметры

sigma

[in] Параметр кривизны функции принадлежности.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const double  x      // аргумент
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

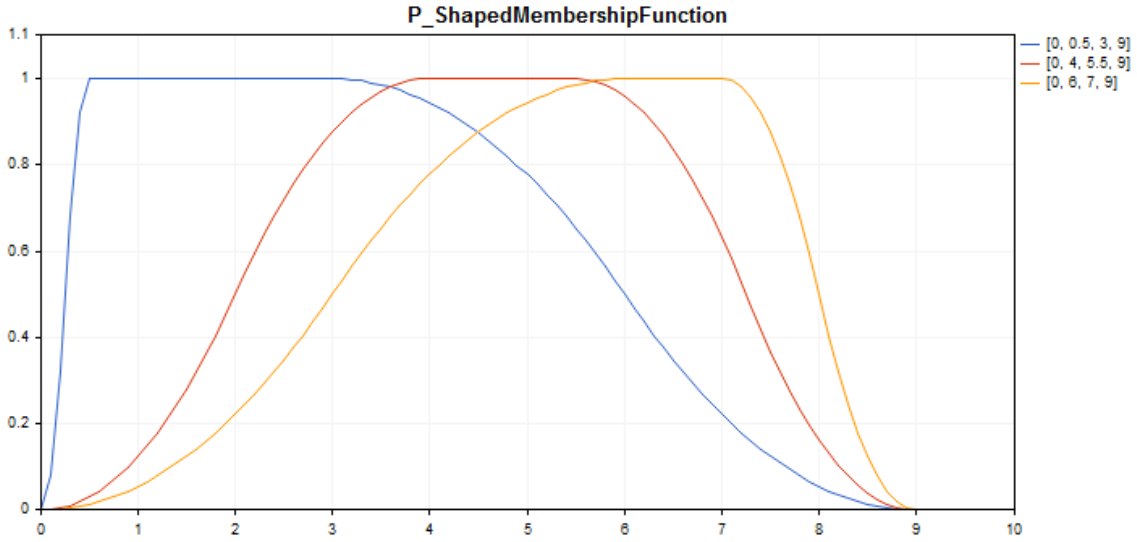
© 2000-2025, MetaQuotes Ltd.

CP_ShapedMembershipFunction

Класс для реализации пи-подобной функции принадлежности с параметрами A, B, C и D.

Описание

Пи-подобная функция принадлежности имеет вид криволинейной трапеции. Функция применяется для задания асимметричных функций принадлежности с плавным переходом от пессимистической к оптимистической оценке нечеткого числа.



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CP_ShapedMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CP_ShapedMembershipFunction

Методы класса

Метод класса	Описание
A	Возвращает и устанавливает параметр начала нечеткого множества.
B	Возвращает и устанавливает первый параметр ядра нечеткого множества.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
C	Возвращает и устанавливает второй параметр ядра нечеткого множества.
D	Возвращает и устанавливает параметр конца нечеткого множества.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                                  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_ShapedMembershipFunction 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
```

© 2000-2025, MetaQuotes Ltd.

```
   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 (Метод Get)

Возвращает параметр начала нечеткого множества.

```
double  A()
```

Возвращаемое значение

Параметр начала нечеткого множества.

A (Метод Set)

Устанавливает параметр начала нечеткого множества.

```
void  A(
   const double  a      // параметр начала нечеткого множества
   )
```

Параметры

a

[in] Параметр начала нечеткого множества.

B (Метод Get)

Возвращает первый параметр ядра нечеткого множества.

```
double  B()
```

Возвращаемое значение

Первый параметр ядра нечеткого множества.

B (Метод Set)

Устанавливает первый параметр ядра нечеткого множества.

```
void  B(
   const double  b      // значение первого параметра ядра нечеткого множества
```

© 2000-2025, MetaQuotes Ltd.

```
   )
```

Параметры

b

[in] Первый параметр ядра нечеткого множества.

C (Метод Get)

Возвращает второй параметр ядра нечеткого множества.

```
double  C()
```

Возвращаемое значение

Второй параметр ядра нечеткого множества.

C (Метод Set)

Устанавливает второй параметр ядра нечеткого множества.

```
void  C(
   const double  c      // значение второго параметра ядра нечеткого множества
   )
```

Параметры

c

[in] Второй параметр ядра нечеткого множества.

D (Метод Get)

Возвращает параметр конца нечеткого множества.

```
double  D()
```

Возвращаемое значение

Значение параметра конца нечеткого множества.

D (Метод Set)

Устанавливает параметр конца нечеткого множества.

```
void  D(
   const double  d      // значение параметра конца нечеткого множества
   )
```

Параметры

d

[in] Значение параметра конца нечеткого множества.

© 2000-2025, MetaQuotes Ltd.

GetValue

Расчитывает значение функции принадлежности по указанному аргументу

```
double  GetValue(
   const double  x
   )
```

Параметры

x

[in] аргумент функции принадлежности

Возвращаемое значение

значение функции принадлежности

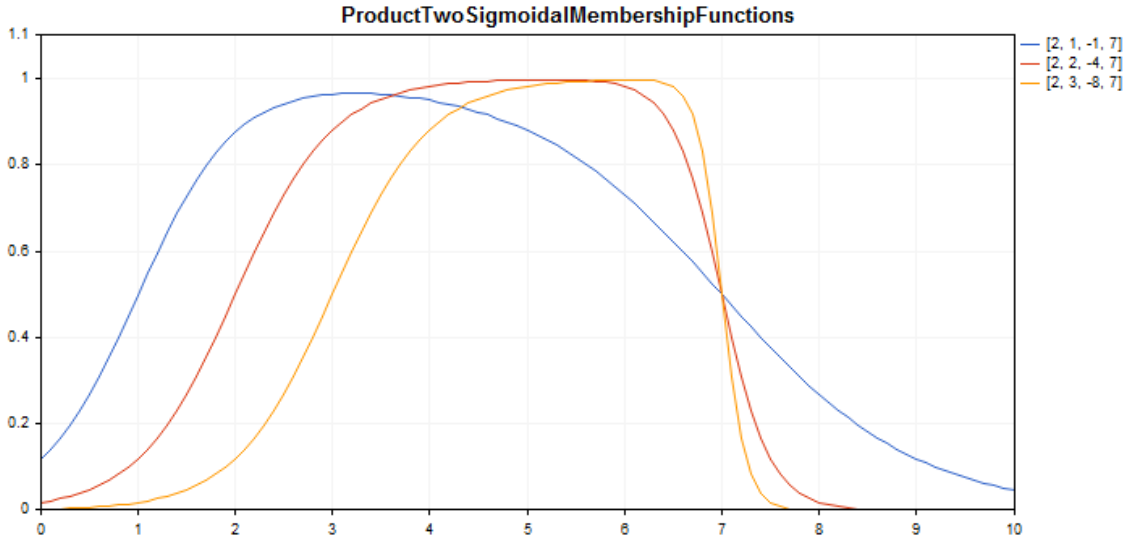
© 2000-2025, MetaQuotes Ltd.

CProductTwoSigmoidalMembershipFunction

Класс для реализации функции принадлежности в виде произведения двух сигмоидных функций с параметрами A1, A2, C1, C2.

Описание

Произведение двух сигмоидных функций принадлежности применяется для задания гладких асимметричных функций. С его помощью можно создавать функции принадлежности со значениями, равными 1, начиная с некоторого значения аргумента. Подобные функции подходят, если необходимо задать лингвистические термы типа "короткий" или "длинный".



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CProductTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CProductTwoSigmoidalMembershipFunctions

Методы класса

Метод класса	Описание
A1	Возвращает и устанавливает коэффициент крутизны первой функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
A2	Возвращает и устанавливает коэффициент крутизны второй функции принадлежности.
C1	Возвращает параметр координаты перегиба первой функции принадлежности.
C2	Возвращает параметр координаты перегиба второй функции принадлежности.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                      ProductTwoSigmoidalMembershipFunctions.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"
#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");
```

© 2000-2025, MetaQuotes Ltd.

```
   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 (Метод Get)

Возвращает коэффициент крутизны первой функции принадлежности.

```
double  A1()
```

Возвращаемое значение

Коэффициент крутизны первой функции принадлежности.

A1 (Метод Set)

Устанавливает коэффициент крутизны первой функции принадлежности.

```
void  A1(
   const double  a1      // коэффициент крутизны первой функции принадлежности
   )
```

Параметры

a1

[in] Коэффициент крутизны первой функции принадлежности.

A2 (Метод Get)

Возвращает коэффициент крутизны второй функции принадлежности.

```
double  A2()
```

Возвращаемое значение

Коэффициент крутизны второй функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

A2 (Метод Set)

Устанавливает коэффициент крутизны второй функции принадлежности.

```
void  A2(
   const double  a2      // коэффициент крутизны второй функции принадлежности
   )
```

Параметры

a2

[in] Коэффициент крутизны второй функции принадлежности.

C1 (Метод Get)

Возвращает параметр координаты перегиба первой функции принадлежности.

```
double  C1()
```

Возвращаемое значение

Координата перегиба первой функции принадлежности.

C1 (Метод Set)

Устанавливает координату перегиба первой функции принадлежности.

```
void  C1(
   const double  c1      // координата перегиба первой функции принадлежности
   )
```

Параметры

c1

[in] Координата перегиба первой функции принадлежности.

C2 (Метод Get)

Возвращает параметр координаты перегиба второй функции принадлежности.

```
double  C2()
```

Возвращаемое значение

Координата перегиба второй функции принадлежности.

C2 (Метод Set)

Устанавливает координату перегиба второй функции принадлежности.

```
void  C2(
   const double  c2      // координата перегиба второй функции принадлежности
   )
```

© 2000-2025, MetaQuotes Ltd.

Параметры

c2

[in] Координата перегиба второй функции принадлежности.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

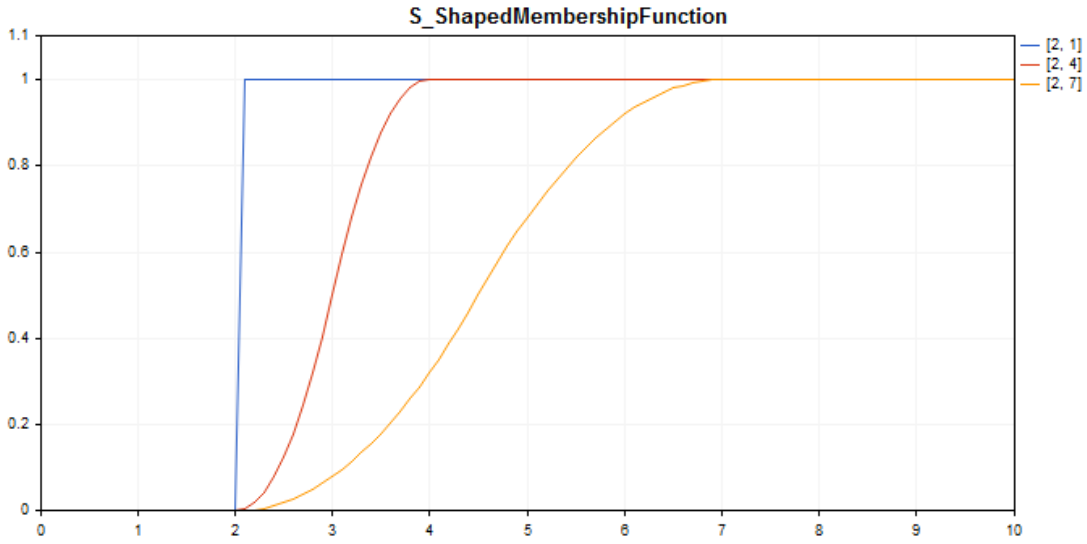
CS_ShapedMembershipFunction

Класс для реализации S-подобной функции принадлежности с параметрами A и B.

Описание

Функция задает S-подобную двухпараметрическую функцию принадлежности. Это неубывающая функция, принимающая значения от 0 до 1. Параметры A и B определяют интервал, внутри которого функция возрастает по нелинейной траектории от 0 до 1.

С помощью этой функции представлены нечеткие множества типа "очень высокий" (то есть, задаются неубывающие функции принадлежности с насыщением).



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CS_ShapedMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CS_ShapedMembershipFunction

Методы класса

Метод класса	Описание
A	Возвращает и устанавливает параметр начала интервала возрастания.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
B	Возвращает и устанавливает первый параметр ядра нечеткого множества.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                                    S_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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_ShapedMembershipFunction 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);
```

© 2000-2025, MetaQuotes Ltd.

```
   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 (Метод Get)

Возвращает параметр начала интервала возрастания.

```
double  A()
```

Возвращаемое значение

Параметр начала интервала возрастания.

A (Метод Set)

Устанавливает параметр начала интервала возрастания.

```
void  A(
   const double  a      // параметр начала интервала возрастания
   )
```

Параметры

a

[in] Параметр начала интервала возрастания.

B (Метод Get)

Возвращает первый параметр ядра нечеткого множества.

```
double  B()
```

Возвращаемое значение

Первый параметр ядра нечеткого множества.

B (Метод Set)

Устанавливает первый параметр ядра нечеткого множества.

```
void  B(
   const double  b      // первый параметр ядра нечеткого множества
   )
```

© 2000-2025, MetaQuotes Ltd.

Параметры

b

[in] Первый параметр ядра нечеткого множества.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

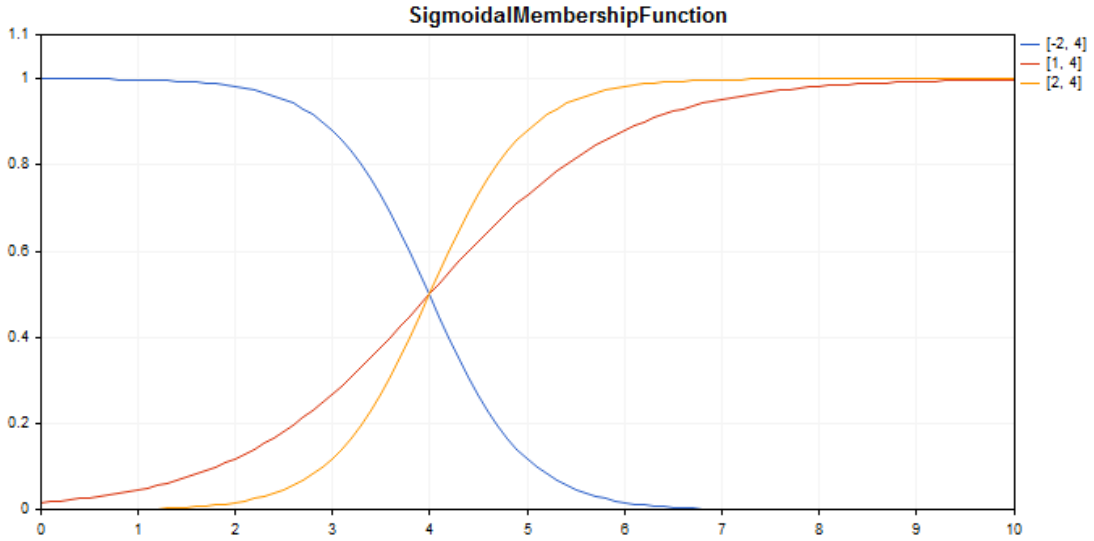
© 2000-2025, MetaQuotes Ltd.

CSigmoidalMembershipFunction

Класс для реализации сигмоидной функции принадлежности с параметрами A и C.

Описание

Сигмоидная функция применяется для задания монотонных функций принадлежности. С ее помощью можно создавать функции принадлежности со значениями, равными 1, начиная с некоторого значения аргумента. Подобные функции подходят, если необходимо задать лингвистические термы типа "короткий" или "длинный".



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CSigmoidalMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CSigmoidalMembershipFunction

Методы класса

Метод класса	Описание
A	Возвращает и устанавливает коэффициент крутизны функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
C	Возвращает и устанавливает параметр координаты перегиба функции принадлежности.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                                  SigmoidalMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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 SigmoidalMembershipFunction2(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);
```

© 2000-2025, MetaQuotes Ltd.

```
   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 (Метод Get)

Возвращает коэффициент крутизны функции принадлежности.

```
double  A()
```

Возвращаемое значение

Коэффициент крутизны функции принадлежности.

A (Метод Set)

Устанавливает коэффициент крутизны функции принадлежности.

```
void  A(
   const double  a      // коэффициент крутизны функции принадлежности
   )
```

Параметры

a

[in] Коэффициент крутизны функции принадлежности.

C (Метод Get)

Возвращает параметр координаты перегиба функции принадлежности.

```
double  C()
```

Возвращаемое значение

Координата перегиба функции принадлежности.

C (Метод Set)

Устанавливает координату перегиба функции принадлежности.

```
void  C(
   const double  c      // координата перегиба функции принадлежности
   )
```

© 2000-2025, MetaQuotes Ltd.

Параметры

c

[in] Координата перегиба функции принадлежности.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

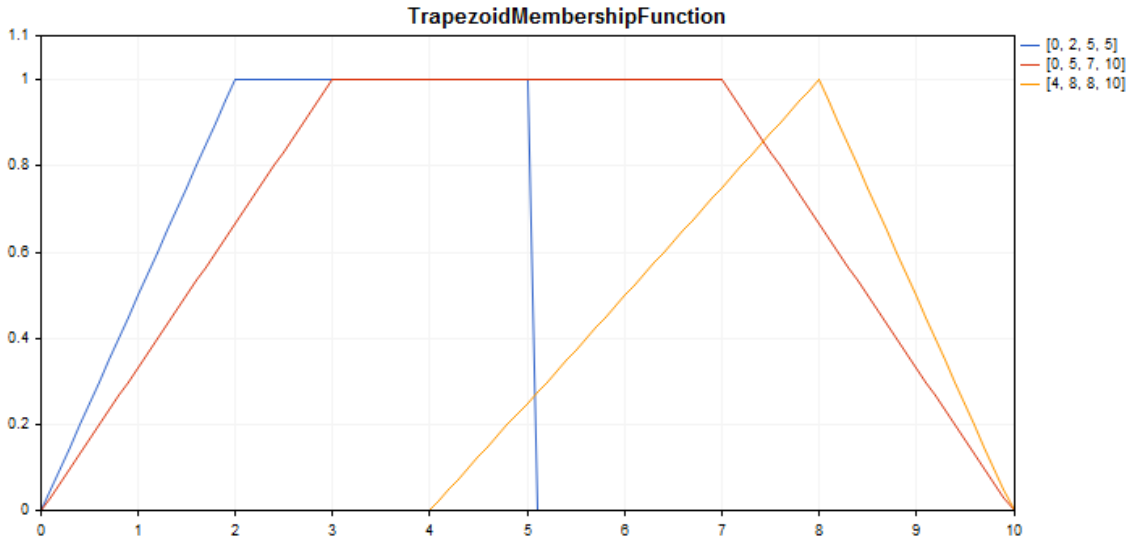
CTrapezoidMembershipFunction

Класс для реализации трапециевидной функции принадлежности с параметрами X1, X2, X3 и X4.

Описание

Функция формируется с использованием кусочно-линейной аппроксимации. Это обобщение треугольной функции, она позволяет задавать ядро нечеткого множетсва в виде интервала. Такая функция принадлежности делает возможной удобную интерпретацию оптимистической/пессимистической оценки.

Применяется для задания асимметричных функций принадлежности переменных, наиболее возможные значения которых принимаются на некотором интервале.



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CTrapezoidMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CTrapezoidMembershipFunction

Методы класса

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
X1	Возвращает и устанавливает значение первой точки на абсциссе.
X2	Возвращает и устанавливает значение второй точки на абсциссе.
X3	Возвращает и устанавливает значение третьей точки на абсциссе.
X4	Возвращает и устанавливает значение четвертой точки на абсциссе.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                                  TrapezoidMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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 TrapezoidMembershipFunction3(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);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 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 (Метод Get)

Возвращает значение первой точки на абсциссе.

```
double  X1()
```

Возвращаемое значение

Значение первой точки на абсциссе.

X1 (Метод Set)

Устанавливает значение первой точки на абсциссе.

```
void  X1(
   const double  x1      // значение первой точки на абсциссе
   )
```

Параметры

x1

[in] Значение первой точки на абсциссе.

X2 (Метод Get)

Возвращает значение второй точки на абсциссе.

```
double  X2()
```

Возвращаемое значение

Значение второй точки на абсциссе.

© 2000-2025, MetaQuotes Ltd.

X2 (Метод Set)

Устанавливает значение второй точки на абсциссе.

```
void  X2(
   const double  x2      // значение второй точки на абсциссе
   )
```

Параметры

x2

[in] Значение второй точки на абсциссе.

X3 (Метод Get)

Возвращает значение третьей точки на абсциссе.

```
double  X3()
```

Возвращаемое значение

Значение третьей точки на абсциссе.

X3 (Метод Set)

Устанавливает значение третьей точки на абсциссе.

```
void  X3(
   const double  x3      // значение третьей точки на абсциссе
   )
```

Параметры

x3

[in] Значение третьей точки на абсциссе.

X4 (Метод Get)

Возвращает значение четвертой точки на абсциссе.

```
double  X4()
```

Возвращаемое значение

Значение четвертой точки на абсциссе.

X4 (Метод Set)

Устанавливает значение четвертой точки на абсциссе.

```
void  X4(
   const double  x4      // значение четвертой точки на абсциссе
   )
```

© 2000-2025, MetaQuotes Ltd.

Параметры

x4

[in] Значение четвертой точки на абсциссе.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

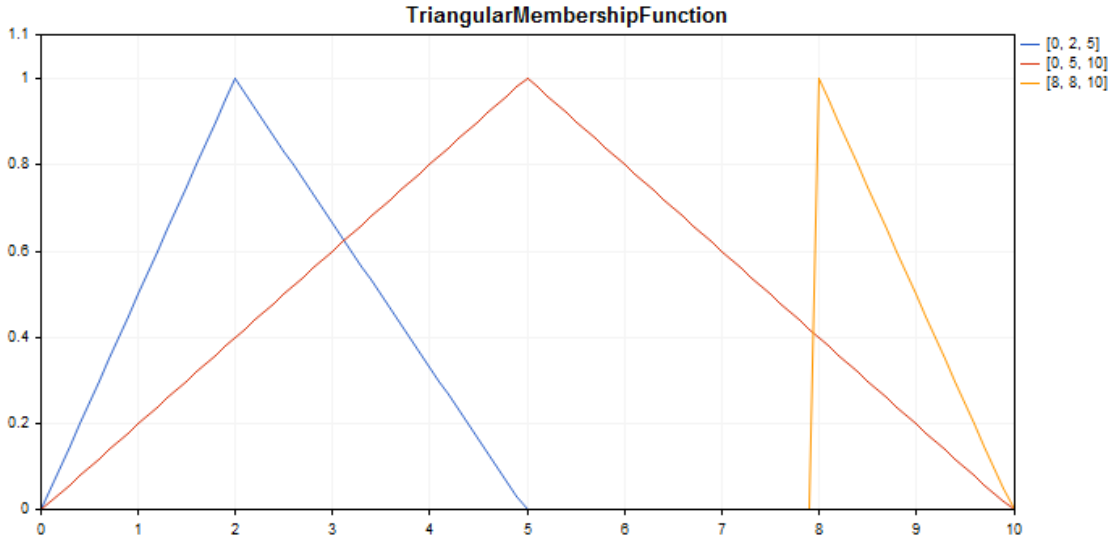
© 2000-2025, MetaQuotes Ltd.

CTriangularMembershipFunction

Класс для реализации треугольной функции принадлежности с параметрами X1, X2, X3.

Описание

Функция задает функцию принадлежности в виде треугольника. Простая и наиболее часто применяемая функция принадлежности.



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CTriangularMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CTriangularMembershipFunction

Методы класса

Метод класса	Описание
X1	Возвращает значение первой точки на абсциссе.
X2	Возвращает значение второй точки на абсциссе.
X3	Возвращает значение третьей точки на абсциссе.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
ToNormalMF	Преобразует треугольную функцию принадлежности в гауссову.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                                 TriangularMembershipFunction.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"
#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]
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 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 (Метод Get)

Возвращает значение первой точки на абсциссе.

```
double  X1()
```

Возвращаемое значение

Значение первой точки на абсциссе.

X1 (Метод Set)

Устанавливает значение первой точки на абсциссе.

```
void  X1(
   const double  x1      // значение первой точки на абсциссе
   )
```

Параметры

x1

[in] Значение первой точки на абсциссе.

X2 (Метод Get)

Возвращает значение второй точки на абсциссе.

```
double  X2()
```

Возвращаемое значение

Значение второй точки на абсциссе.

X2 (Метод Set)

Устанавливает значение второй точки на абсциссе.

```
void  X2(
```

© 2000-2025, MetaQuotes Ltd.

```
   const double  x2      // значение второй точки на абсциссе
   )
```

Параметры

x2

[in] Значение второй точки на абсциссе.

X3 (Метод Get)

Возвращает значение третьей точки на абсциссе.

```
double  X3()
```

Возвращаемое значение

Значение третьей точки на абсциссе.

X3 (Метод Set)

Устанавливает значение третьей точки на абсциссе.

```
void  X3(
   const double  x3      // значение третьей точки на абсциссе
   )
```

Параметры

x3

[in] Значение третьей точки на абсциссе.

ToNormalMF

Преобразует треугольную функцию принадлежности в Гауссовскую.

```
CNormalMembershipFunction*  ToNormalMF()
```

Возвращаемое значение

Указатель на Гауссовскую функцию принадлежности.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

© 2000-2025, MetaQuotes Ltd.

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

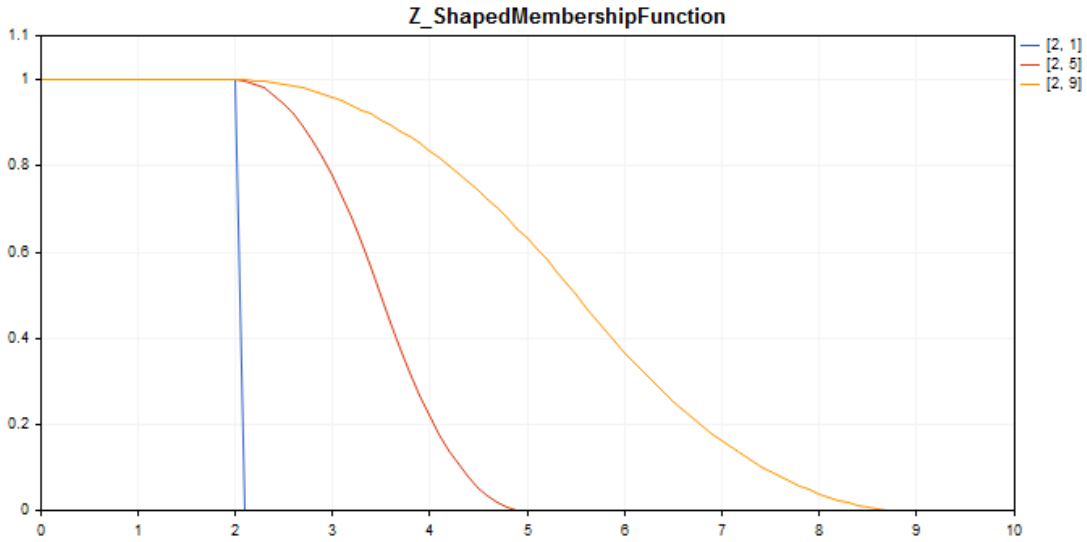
CZ_ShapedMembershipFunction

Класс для реализации z-подобной функции принадлежности с параметрами A и B.

Описание

Функция задает z-подобную двухпараметрическую функцию принадлежности. Это невозрастающая функция принадлежности, принимающая значения от 1 до 0. Параметры функции определяют интервал, внутри которого функция нелинейно убывает от 0 до 1.

С помощью функции могут быть представлены нечеткие множества типа "очень низкий". То есть, она задает невозрастающие функции принадлежности с насыщением.



Пример кода для построения этого графика приведен ниже.

Декларация

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

CZ_ShapedMembershipFunction

Методы класса

Метод класса	Описание
A	Возвращает и устанавливает параметр начала интервала убывания.

© 2000-2025, MetaQuotes Ltd.

Метод класса	Описание
B	Возвращает и устанавливает параметр конца интервала убывания.
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Пример

```
//+------------------------------------------------------------------+
//|                                  Z_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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_ShapedMembershipFunction 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);
```

© 2000-2025, MetaQuotes Ltd.

```
   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 (Метод Get)

Возвращает параметр начала интервала убывания.

```
double  A()
```

Возвращаемое значение

Параметр начала интервала убывания.

A (Метод Set)

Устанавливает параметр начала интервала убывания.

```
void  A(
   const double  a      // параметр начала интервала убывания
   )
```

Параметры

a

[in] Параметр начала интервала убывания.

B (Метод Get)

Возвращает параметр конца интервала убывания.

```
double  B()
```

Возвращаемое значение

Параметр конца интервала убывания.

B (Метод Set)

Устанавливает параметр конца интервала убывания.

```
void  B(
   const double  b      // параметр конца интервала убывания
   )
```

© 2000-2025, MetaQuotes Ltd.

Параметры

b

[in] Параметр конца интервала убывания.

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

IMembershipFunction

Базовый класс для всех классов функций принадлежности.

Декларация

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Заголовок

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Иерархия наследования

CObject

IMembershipFunction

Прямые потомки

CCompositeMembershipFunction, CConstantMembershipFunction, CDifferencTwoSigmoidalMembershipFunction, CGeneralizedBellShapedMembershipFunction, CNormalCombinationMembershipFunction, CNormalMembershipFunction, CP_ShapedMembershipFunction, CProductTwoSigmoidalMembershipFunctions, CS_ShapedMembershipFunction, CSigmoidalMembershipFunction, CTrapezoidMembershipFunction, CTriangularMembershipFunction, CZ_ShapedMembershipFunction

Методы класса

Метод класса	Описание
GetValue	Рассчитывает значение функции принадлежности по указанному аргументу.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

GetValue

Рассчитывает значение функции принадлежности по указанному аргументу.

```
double  GetValue(
   const  x      // аргумент функции принадлежности
   )
```

Параметры

x

[in] Аргумент функции принадлежности.

Возвращаемое значение

Значение функции принадлежности.

© 2000-2025, MetaQuotes Ltd.

Правила для нечетких систем

Нечеткая система (система нечеткого логического вывода) — это получение заключения в виде нечеткого множества, соответствующего текущим значениям входов, с использованием совокупности **нечетких правил** и нечетких операций.

Нечеткие правила определяют взаимосвязь между входами и выходами исследуемого объекта. Количество правил в системе не ограничено. Обобщенный формат нечетких правил таков:

если *условие (посылка) правила*, **то** *заключение правила*.

Условие правила характеризует текущее состояние объекта. *Заключение правила* характеризует то, как условие повлияет на объект.

Класс правил для нечетких систем	Описание
CMamdaniFuzzyRule	Класс для реализации нечеткого логического правила для алгоритма Мамдани
CSugenoFuzzyRule	Класс для реализации нечеткого логического правила для алгоритма Сугено
CSingleCondition	Класс задает нечеткое условие, выраженное парой "Нечеткая переменная — Нечеткий терм".
CConditions	Класс задает набор нечетких условий, связанных оператором.
CGenericFuzzyRule	Базовый класс для реализации обоих типов нечетких правил.

© 2000-2025, MetaQuotes Ltd.

CMamdaniFuzzyRule

Нечеткий логический вывод Мамдани — один из двух основных типов нечетких систем. Значения выходной переменной в нем задаются нечеткими термами.

Описание

Нечеткое логическое правило для алгоритма Мамдани можно описать следующим выражением:

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

где:

- X = (X1, X2, X3 ... Xn) — вектор входных переменных;
- Y — выходная переменная;
- a = (a1, a2, a3 ... an) — вектор значений входных переменных;
- d — значение выходной переменной;
- W — вес правила.

Декларация

```
class CMamdaniFuzzyRule : public CGenericFuzzyRule
```

Заголовок

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Иерархия наследования

CObject

IParsableRule

CGenericFuzzyRule

CMamdaniFuzzyRule

Методы класса

Метод класса	Описание
Conclusion	Возвращает и устанавливает заключение нечёткого правила Мамдани.
Weight	Возвращает и устанавливает вес нечёткого правила Мамдани.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

© 2000-2025, MetaQuotes Ltd.

Conclusion (Метод Get)

Возвращает заключение нечёткого правила Мамдани.

```
CSingleConditon*  Conclusion()
```

Возвращаемое значение

Заключение нечеткого правила Мамдани.

Conclusion (Метод Set)

Устанавливает заключение нечёткого правила Мамдани.

```
void  Conclusion(
   CSingleConditon*  value      // заключение нечёткого правила Мамдани
   )
```

Параметры

value

[in] Заключение нечёткого правила Мамдани.

Weight (Метод Get)

Возвращает вес нечёткого правила Мамдани.

```
double  Weight()
```

Возвращаемое значение

Вес нечёткого правила Мамдани.

Weight (Метод Set)

Устанавливает вес нечёткого правила Мамдани.

```
void  Weight(
   const double  value      // вес нечёткого правила Мамдани
   )
```

Параметры

value

[in] Вес нечёткого правила Мамдани.

© 2000-2025, MetaQuotes Ltd.

CSugenoFuzzyRule

Нечеткий логический вывод Сугено — один из двух основных типов нечетких систем. Значения выходной переменной в нем задаются как линейная комбинация входных переменных.

Описание

Отличие от правила Мамдани заключается в том, что значение выходной переменной задается не нечетким термом, а линейной функцией от входных переменных. Нечеткое логическое правило для алгоритма Сугено можно описать следующим выражением:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

где:

- X = (X1, X2, X3 ... Xn) — вектор входных переменных;
- Y — выходная переменная;
- a = (a1, a2, a3 ... an) — вектор значений входных переменных;
- b = (b1, b2, b3 ... bn) — коэффициент при свободном члене в линейной функции для выходного значения
- W — вес правила.

Декларация

```
class CSugenoFuzzyRule : public CGenericFuzzyRule
```

Заголовок

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Иерархия наследования

CObject

IParsableRule

CGenericFuzzyRule

CSugenoFuzzyRule

Методы класса

Метод класса	Описание
Conclusion	Возвращает и устанавливает заключение нечёткого правила Сугено.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

© 2000-2025, MetaQuotes Ltd.

Conclusion (Метод Get)

Возвращает заключение нечёткого правила Сугено.

```
CSingleConditon*  Conclusion()
```

Возвращаемое значение

Заключение нечёткого правила Сугено.

Conclusion (Метод Set)

Устанавливает заключение нечёткого правила Сугено.

```
void  Conclusion(
   CSingleCondition*  value      // заключение нечёткого правила Сугено
   )
```

Параметры

value

[in] Заключение нечёткого правила Сугено.

© 2000-2025, MetaQuotes Ltd.

CSingleCondition

Класс задает нечеткое условие, выраженное парой "Нечеткая переменная — Нечеткий терм".

Описание

По нечеткому условию одной переменной соответствует один терм. Нечеткое условие можно описать следующим выражением: *X is a*,

где:

- X — нечеткая переменная;
- a — значение нечеткой переменной (нечеткий терм).

Декларация

```
class CSingleCondition : public ICondition
```

Заголовок

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Иерархия наследования

CObject
  ICondition
    CSingleCondition

Прямые потомки

CFuzzyCondition

Методы класса

Метод класса	Описание
Not	Возвращает и устанавливает флаг, указывающий на то, применить ли к этому условию отрицание.
Term	Возвращает и устанавливает нечёткий терм для данного условия.
Var	Возвращает и устанавливает нечёткую переменную для данного условия.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Not (Метод Get)

Возвращает флаг, указывающий на то, применить ли к этому условию отрицание.

```
bool Not()
```

© 2000-2025, MetaQuotes Ltd.

Возвращаемое значение

Значение флага.

Not (Метод Set)

Устанавливает флаг, указывающий на то, применить ли к этому условию отрицание.

```
void  Not(
   bool  not      // значение флага
   )
```

Параметры

not

[in] Значение флага.

Term (Метод Get)

Возвращает нечёткий терм для данного условия.

```
INamedValue* Term()
```

Возвращаемое значение

Нечёткий терм для данного условия.

Term (Метод Set)

Устанавливает нечёткий терм для данного условия.

```
void  Term(
   INamedValue*&  value      // нечёткий терм для данного условия
   )
```

Параметры

value

[in] Нечёткий терм для данного условия.

Var (Метод Get)

Возвращает нечёткую переменную для данного условия.

```
INamedVariable* Var()
```

Возвращаемое значение

Нечёткая переменная для данного условия.

Var (Метод Set)

Устанавливает нечёткую переменную для данного условия

© 2000-2025, MetaQuotes Ltd.

```
void  Var(
   INamedVariable*&  value      // нечёткая переменная для данного условия
   )
```

Параметры

value

[in] нечёткая переменная.

© 2000-2025, MetaQuotes Ltd.

CConditions

Класс задает набор нечетких условий, связанных оператором.

Описание

Набор нечетких условий, связанных оператором, может быть описан, например, следующим выражением:

$$(X_1 \; is \; a_1) \wedge (X_2 \; is \; a_2) \wedge \ldots \wedge (X_n \; is \; a_n)$$

где:

- X = (X1, X2, X3 ... Xn) — вектор входных переменных;
- a = (a1, a2, a3 ... an) — вектор значений входных переменных.

В данном примере использован оператор *and* (и). Также в этом классе доступен оператор *or* (или).

Декларация

```
class CConditions : public ICondition
```

Заголовок

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Иерархия наследования

CObject
  ICondition
    CConditions

Методы класса

Метод класса	Описание
ConditionsList	Возвращает список всех условий
Not	Возвращает и устанавливает флаг, указывающий на то, применять ли к этим условиям отрицание
Op	Возвращает и устанавливает тип оператора связки условий.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

ConditionsList

Возвращает список всех условий.

© 2000-2025, MetaQuotes Ltd.

```
CList*  ConditionsList()
```

Возвращаемое значение

Список всех условий.

Not (Метод Get)

Возвращает флаг, указывающий на то, применять ли к этим условиям отрицание.

```
bool Not()
```

Возвращаемое значение

Значение флага.

Not (Метод Set)

Устанавливает флаг указывающий на то, применять ли к этим условиям отрицание

```
void  Not(
   bool  not      // значение флага
   )
```

Параметры

not

[in] Значение флага.

Op (Метод Get)

Возвращает тип оператора связки условий. Доступны два оператора: *and* (и) и *or* (или).

```
OperatorType Op()
```

Возвращаемое значение

Тип оператора связки условий.

Op (Метод Set)

Устанавливает оператор связки условий. Доступны два оператора: *and* (и) и *or* (или).

```
void  Op(
   OperatorType  op     // тип оператора связки условий
   )
```

Параметры

op

[in] Тип оператора связки условий.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

CGenericFuzzyRule

Базовый класс для обоих типов нечетких правил.

Декларация

```
class CGenericFuzzyRule : public IParsableRule
```

Заголовок

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Иерархия наследования

CObject

IParsableRule

CGenericFuzzyRule

Прямые потомки

CMamdaniFuzzyRule, CSugenoFuzzyRule

Методы класса

Метод класса	Описание
Conclusion	Возвращает и устанавливает заключение нечёткого правила
Condition	Возвращает и устанавливает условие (набор условий) 'if' для нечёткого правила
CreateCondition	Создаёт условие для нечёткого правила по заданным параметрам

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Conclusion (Метод Get)

Возвращает заключение нечёткого правила.

```
CSingleConditon*  Conclusion()
```

Возвращаемое значение

Заключение нечеткого правила.

Conclusion (Метод Set)

Устанавливает заключение нечёткого правила.

```
virtual void  Conclusion(
```

© 2000-2025, MetaQuotes Ltd.

```
   CSingleConditon*  value      // заключение нечёткого правила
   )
```

Параметры

value

[in] Заключение нечёткого правила.

Condition (Метод Get)

Возвращает условие (набор условий) 'if' для нечёткого правила.

```
CConditons*  Condition()
```

Возвращаемое значение

Нечёткое условие (набор условий).

Condition (Метод Set)

Устанавливает условие (набор условий) 'if' для нечёткого правила.

```
void  Condition(
   CConditons*  value      // условие (набор условий) "if" для нечёткого правила
   )
```

Параметры

value

[in] Нечёткое условие (набор условий).

CreateCondition

Создаёт условие для нечёткого правила по заданным параметрам.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,      // нечёткая переменная
   CFuzzyTerm*      term,     // нечёткий терм
   )
```

Параметры

var

[in] Нечёткая переменная.

term

[in] Нечёткий терм.

Возвращаемое значение

Состояние нечёткого правила.

© 2000-2025, MetaQuotes Ltd.

CreateCondition

Создаёт условие для нечёткого правила по заданным параметрам.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,       // нечёткая переменная
   CFuzzyTerm*      term,      // нечёткий терм
   bool             not,       // флаг, указывающий, нужно ли применять к условию отри
   )
```

Параметры

var

[in] Нечёткая переменная.

term

[in] Нечёткий терм.

not

[in] Флаг, указывающий, нужно ли применять к условию отрицание.

Возвращаемое значение

Состояние нечёткого правила.

CreateCondition

Создаёт условие для нечёткого правила по заданным параметрам.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable* var,       // нечёткая переменная
   CFuzzyTerm*     term,      // нечёткий терм
   bool            not,       // флаг, указывающий, нужно ли применять к условию отриц
   HedgeType       hedge      // тип связки условия
   )
```

Параметры

var

[in] Нечёткая переменная.

term

[in] Нечёткий терм.

not

[in] Флаг, указывающий, нужно ли применять к условию отрицание.

hedge

[in] Тип связки условия.

Возвращаемое значение

Состояние нечёткого правила.

© 2000-2025, MetaQuotes Ltd.

Переменные для нечетких систем

В нечётких системах применяются **нечёткие (лингвистические) переменные**. Это такие переменные, значениями которых могут быть слова или словосочетания некоторого естественного или искусственного языка.

Лингвистические переменные составляют нечеткие множества. Характер и количество нечётких переменных при определении нечётких множеств изменяются для каждой отдельной задачи.

Класс	Описание
CFuzzyVariable	Класс, с помощью которого создаются нечеткие переменные общего вида.
CSugenoVariable	Класс, с помощью которого создаются нечеткие переменные типа Сугено.

© 2000-2025, MetaQuotes Ltd.

CFuzzyVariable

Класс, с помощью которого создаются нечеткие переменные общего вида.

Описание

Нечеткая переменная в нашем случае задается следующими параметрами:

- максимальное значение переменной;
- минимальное значение переменной;
- имя нечеткой переменной;
- терм-множество (множество всех возможных значений, которые способна принимать лингвистическая переменная).

Декларация

```
class CFuzzyVariable : public CNamedVariableImpl
```

Заголовок

```
#include <Math\Fuzzy\fuzzyvariable.mqh>
```

Иерархия наследования

CObject
INamedValue
INamedVariable
CNamedVariableImpl
CFuzzyVariable

Методы класса

Метод класса	Описание
AddTerm	Добавляет один нечеткий терм к нечеткой переменной.
GetTermByName	Получает нечеткий терм по заданному имени.
Max	Возвращает и устанавливает максимальное значение для нечеткой переменной.
Min	Возвращает и устанавливает минимальное значение для нечеткой переменной.
Terms	Возвращает и устанавливает список нечетких термов для данной нечеткой переменной.
Values	Возвращает и устанавливает список нечетких термов для данной нечеткой переменной.

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CNamedVariableImpl

Name, Name

AddTerm

Добавляет один нечёткий терм к нечёткой переменной.

```
void  AddTerm(
   CFuzzyTerm*&  term      // нечеткий терм
   )
```

Параметры

term

[in] Нечёткий терм.

GetTermByName

Получает нечёткий терм по заданному имени.

```
CFuzzyTerm*  GetTermByName(
   const string  name      // имя нечёткого терма
   )
```

Параметры

name

[in] Имя нечёткого терма.

Возвращаемое значение

Нечёткий терм с заданным именем.

Max (Метод Get)

Возвращает максимальное значение для нечёткой переменной.

```
double  Max()
```

Возвращаемое значение

Максимальное значение для нечеткой переменной.

Max (Метод Set)

Устанавливает максимальное значение для нечёткой переменной.

```
void  Max(
   const double  max      // максимальное значение для нечёткой переменной
```

© 2000-2025, MetaQuotes Ltd.

```
   )
```

Параметры

max

[in] Максимальное значение для нечёткой переменной.

Min (Метод Get)

Возвращает минимальное значение для нечёткой переменной.

```
double  Min()
```

Возвращаемое значение

Минимальное значение для нечеткой переменной.

Max (Метод Set)

Устанавливает минимальное значение для нечёткой переменной.

```
void  Min(
   const double  min      // минимальное значение для нечёткой переменной
   )
```

Параметры

min

[in] Минимальное значение для нечёткой переменной.

Terms (Метод Get)

Возвращает список нечётких термов для данной нечёткой переменной.

```
CList*  Terms()
```

Возвращаемое значение

Список нечётких термов для данной нечёткой переменной.

Terms (Метод Set)

Устанавливает список нечётких термов для данной нечёткой переменной.

```
void  Terms(
   CList*&  terms      // список нечётких термов для данной переменной
   )
```

Параметры

terms

[in] список нечётких термов для данной нечёткой переменной.

© 2000-2025, MetaQuotes Ltd.

Values

Возвращает список нечётких термов для данной нечёткой переменной.

```
CList*  Values()
```

Возвращаемое значение

Список нечётких термов для данной переменной.

© 2000-2025, MetaQuotes Ltd.

CSugenoVariable

Класс, с помощью которого создаются нечеткие переменные типа Сугено.

Описание

Нечеткая переменная типа Сугено отличается от лингвистической переменной общего типа тем, что задается не терм-множеством, а набором линейных функций.

Декларация

```
class CSugenoVariable : public CNamedVariableImpl
```

Заголовок

```
#include <Math\Fuzzy\sugenovariable.mqh>
```

Иерархия наследования

CObject
  INamedValue
    INamedVariable
      CNamedVariableImpl
        CSugenoVariable

Методы класса

Метод класса	Описание
Functions	Возвращает список линейных функций нечёткой переменной Сугено.
GetFuncByName	Возвращает линейную функцию по заданному имени.
Values	Возвращает список линейных функций нечёткой переменной Сугено.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CNamedVariableImpl

Name, Name

Functions

Возвращает список линейных функций нечёткой переменной Сугено.

```
CList*  Functions()
```

Возвращаемое значение

© 2000-2025, MetaQuotes Ltd.

Список линейных функций.

GetFuncByName

Возвращает линейную функцию по заданному имени.

```
ISugenoFunction*  GetFuncByName(
   const string  name       // имя линейной функции
   )
```

Параметры

name

[in] Имя линейной функции.

Возвращаемое значение

Линейная функция с заданным именем.

Values

Возвращает список линейных функций нечёткой переменной Сугено.

```
CList*  Values()
```

Возвращаемое значение

Список линейных функций нечёткой переменной Сугено.

© 2000-2025, MetaQuotes Ltd.

CFuzzyTerm (нечеткие термы)

Класс для реализации нечетких термов.

Описание

Термом (term) называется любой элемент терм-множества. Терм определяется двумя составляющими:

- именем нечеткого терма;
- функцией принадлежности.

Декларация

```
class CFuzzyTerm : public CNamedValueImpl
```

Заголовок

```
#include <Math\Fuzzy\fuzzyterm.mqh>
```

Иерархия наследования

CObject
  INamedValue
    CNamedValueImpl
      CFuzzyTerm

Методы класса

Метод класса	Описание
MembershipFunction	Возвращает функцию принадлежности для данного нечеткого терма.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CNamedValueImpl

Name, Name

© 2000-2025, MetaQuotes Ltd.

MembershipFunction

Возвращает функцию принадлежности для данного нечеткого терма.

```
IMembershipFunction*  MembershipFunction()
```

Возвращаемое значение

Функция принадлежности.

© 2000-2025, MetaQuotes Ltd.

Нечеткие системы

Нечеткая система (*или нечеткая модель*) — математическая модель, в основе вычисления которой лежит нечеткая логика. К построению таких моделей прибегают в случае, когда предмет исследования имеет очень слабую формализацию, и его точное математическое описание слишком сложное или неизвестно.

Ход построения модели можно разделить на три основных этапа:

1. Определение входных и выходных параметров модели.
2. Построение базы знаний.
3. Выбор одного из методов нечеткого логического вывода (Мамдани или Сугено).

От первого этапа непосредственно зависят два других, и именно он определяет будущее функционирование модели.

База знаний (*база правил*) — это совокупность нечетких правил вида: "если, то", определяющих взаимосвязь между входами и выходами исследуемого объекта.

Условие (Condition) правила характеризует текущее состояние объекта, а **заключение (Conclusion)** — то, как это условие повлияет на объект.

Условия и заключения для каждого правила могут быть двух видов:

1. простое (ссылка на Csinglcond) — в нем участвует одна нечеткая переменная;
2. составное (ссылка Cconditions) — участвуют несколько нечетких переменных.

Каждое правило в системе имеет **вес** — значимость правила в модели. Весовые коэффициенты присваиваются правилу в диапазоне [0, 1].

В зависимости от созданной базы знаний для модели определяется система нечеткого логического вывода. **Нечетким логическим выводом** называется получение заключения в виде нечеткого множества, соответствующего текущим значениям входов, с использованием нечеткой базы знаний и нечетких операций. Два основных типа нечеткого логического вывода — Мамдани и Сугено.

© 2000-2025, MetaQuotes Ltd.

Система Мамдани

Значения выходной переменной в системе Мамдани задаются нечеткими термами.

Описание

Нечеткое логическое правило для алгоритма Мамдани можно описать следующим выражением:

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

где:

- X = (X1, X2, X3 ... Xn) — вектор входных переменных;
- Y — выходная переменная;
- a = (a1, a2, a3 ... an) — вектор значений входных переменных;
- d — значение выходной переменной;
- W — вес правила.

Методы класса

Метод класса	Описание
AggregationMethod	Устанавливает тип агрегации условий
Calculate	Производит расчет нечеткого логического вывода для данной системы
DefuzzificationMethod	Устанавливает тип метода дефаззификации
EmptyRule	Создает пустое нечеткое правило Мамдани на основе текущей системы
ImplicationMethod	Устанавливает тип оператора импликации системы
Output	Возвращает список выходных нечётких переменных Мамдани.
OutputByName	Возвращает выходную нечёткую переменную Мамдани по заданному имени.
ParseRule	Создаёт нечёткое правило Мамдани на основе заданной строки.
Rules	Возвращает список нечётких правил Мамдани.

Методы унаследованные от CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

AggregationMethod

Устанавливает тип метода агрегации условий.

```
void  AggregationMethod(
```

© 2000-2025, MetaQuotes Ltd.

```
   AggregationMethod  value      // тип метода агрегации
   )
```

Параметры

value

[in] Тип метода агрегациии условий.

Calculate

Производит расчет нечёткого логического вывода для данной системы.

```
CList*  Calculate(
   CList*  inputValues      // входные данные
   )
```

Параметры

inputValues

[in] Входные данные для расчета.

Возвращаемое значение

Результат расчетов.

DefuzzificationMethod

Устанавливает тип метода дефаззификации.

```
void  DefuzzificationMethod(
   DefuzzificationMethod  value      // тип метода дефаззификации.
   )
```

Параметры

value

[in] Тип метода дефаззификации.

EmptyRule

Создаёт пустое нечёткое правило Мамдани на основе текущей системы.

```
CMamdaniFuzzyRule*  EmptyRule()
```

Возвращаемое значение

Нечёткое правило Мамдани.

ImplicationMethod

Устанавливает тип оператора импликации условий.

```
void  ImplicationMethod(
```

© 2000-2025, MetaQuotes Ltd.

```
   ImplicationMethod  value      // тип оператора импликации
   )
```

Параметры

value

[in] Тип оператора импликации условий.

Output

Возвращает список выходных нечётких переменных Мамдани.

```
CList*  Output()
```

Возвращаемое значение

Список нечётких переменных.

OutputByName

Возвращает выходную нечёткую переменную Мамдани по заданному имени.

```
CFuzzyVariable*  OutputByName(
   const string  name      // имя нечёткой переменной
   )
```

Параметры

name

[in] Имя нечёткой переменной.

Возвращаемое значение

Нечёткая переменная Мамдани с заданным именем.

ParseRule

Создаёт нечёткое правило Мамдани на основе заданной строки.

```
CMamdaniFuzzyRule*  ParseRule(
   const string  rule      // строковое представление нечёткого правила
   )
```

Параметры

rule

[in] Строковое представление нечёткого правила Мамдани.

Возвращаемое значение

Нечёткое правило Мамдани.

© 2000-2025, MetaQuotes Ltd.

Rules

Возвращает список нечётких правил Мамдани.

```
CList*  Rules()
```

Возвращаемое значение

Список нечётких правил Мамдани.

© 2000-2025, MetaQuotes Ltd.

CSugenoFuzzyRule

Нечеткая логическая система Сугено — один из двух основных типов нечетких систем. Значения выходной переменной в нем задаются как линейная комбинация входных переменных.

Описание

Отличие от системы Мамдани заключается в том, что значение выходной переменной задается не нечетким термом, а линейной функцией от входов. Нечеткое логическое правило для алгоритма Сугено можно описать следующим выражением:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

где:

- X = (X1, X2, X3 ... Xn) — вектор входных переменных;
- Y — выходная переменная;
- a = (a1, a2, a3 ... an) — вектор значений входных переменных;
- b = (b1, b2, b3 ... bn) — коэффициент при свободном члене в линейной функции для выходного значения
- W — вес правила.

Методы класса

Метод класса	Описание
Calculate	Производит расчет нечёткого логического вывода для данной системы.
CreateSugenoFunction	Создаёт линейную функцию Сугено для данной системы.
EmptyRule	Создаёт пустое нечёткое правило Сугено на основе текущей системы.
Output	Возвращает список выходных нечётких переменных Сугено.
OutputByName	Возвращает выходную нечёткую переменную Сугено по заданному имени.
ParseRule	Создаёт нечёткое правило Сугено на основе заданной строки.
Rules	Возвращает список нечётких правил.

Методы унаследованные от CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

Calculate

Производит расчет нечёткого логического вывода для данной системы.

© 2000-2025, MetaQuotes Ltd.

```
CList*  Calculate(
   CList*&  inputValues      // входные данные
   )
```

Параметры

inputValues

[in] Входные данные для расчета.

Возвращаемое значение

Результат расчетов.

CreateSugenoFunction

Создаёт линейную функцию Сугено для данной системы.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // имя функции
   const double& coeffs[]       // коэффициенты функции
   )
```

Параметры

name

[in] Имя функции.

coeffs[]

[in] Коэффициенты функции.

Возвращаемое значение

Линейная функция Сугено.

Примечание

Размер массива коэффициентов может быть либо равен количеству входных переменных, либо больше него на единицу. В первом случае свободный член линейной функции Сугено будет равен нулю, а во втором — последнему коэффициенту.

CreateSugenoFunction

Создаёт линейную функцию Сугено для данной системы.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // имя функции
   CList*&       coeffs,        // список пар нечёткая переменная - коэффициент при не
   const double  constValue     // коэффициент при свободном члене функции
   )
```

Параметры

name

[in] Имя функции.

© 2000-2025, MetaQuotes Ltd.

coeffs[]

[in] Коэффициенты функции.

Возвращаемое значение

Линейная функция Сугено.

EmptyRule

Создаёт пустое нечёткое правило Сугено на основе текущей системы.

```
CSugenoFuzzyRule*  EmptyRule()
```

Возвращаемое значение

Нечёткое правило Сугено.

Output

Возвращает список выходных нечётких переменных Сугено.

```
CList*  Output()
```

Возвращаемое значение

Список нечётких переменных.

OutputByName

Возвращает выходную нечёткую переменную Сугено по заданному имени.

```
CSugenoVariable*  OutputByName(
   const string  name      // имя нечеткой переменной
   )
```

Параметры

name

Имя нечёткой переменной.

Возвращаемое значение

Нечёткая переменная Сугено с заданным именем.

ParseRule

Создаёт нечёткое правило Сугено на основе заданной строки.

```
CSugenoFuzzyRule*  ParseRule(
   const string  rule      // строковое представление нечёткого правила Сугено
   )
```

Параметры

© 2000-2025, MetaQuotes Ltd.

rule

[in] Строковое представление нечёткого правила Сугено.

Возвращаемое значение

Нечеткое правило Сугено.

Rules

Возвращает список нечётких правил.

```
CList*  Rules()
```

Возвращаемое значение

Список нечетких правил.

© 2000-2025, MetaQuotes Ltd.

Класс для работы с программами OpenCL

Класс COpenCL является оберткой для удобной работы с функциями OpenCL. Использование GPU в некоторых случаях позволяет значительно увеличить скорость вычислений.

Примеры использования класса для расчетов на числах float и double можно найти в папке MQL5\Scripts\Examples\OpenCL\ в соответствующих подкаталогах, исходные коды самих OpenCL-программ находятся в подкаталогах MQL5\Scripts\Examples\OpenCL\Double\Kernels и MQL5\Scripts\Examples\OpenCL\Float\Kernels .

- MatrixMult.mq5 - пример умножения матриц с использованием глобальной и локальной памяти
- BitonicSort.mq5 - пример параллельной сортировки элементов массива на GPU
- FFT.mq5 - пример расчета быстрого преобразования Фурье
- Wavelet.mq5 - пример расчета вейвлет-преобразования данных при помощи вейвлета Морле.

Исходный код OpenCL рекомендуется писать в отдельных CL-файлах, которые затем можно подключать к MQL5-программе с помощью ресурсных переменных.

Декларация

```
class COpenCL
```

Заголовок

```
#include <OpenCL\OpenCL.mqh>
```

Методы класса

Имя	Описание
BufferCreate	Создает буфер OpenCL по указанному индексу
BufferFree	Удаляет буфер по указанному индексу
BufferFromArray	Создает буфер по указанному индексу из массива значений
BufferRead	Считывает в массив буфер OpenCL по указанному индексу
BufferWrite	Записывает массив данных в буфер по указанному индексу
Execute	Выполняет OpenCL программу по указанному индексу
GetContext	Возвращает хендл контекста OpenCL
GetKernel	Возвращает хендл объекта кернел по указанному индексу
GetKernelName	Возвращает имя объекта кернел по указанному индексу
GetProgram	Возвращает хендл программы OpenCL
Initialize	Инициализирует программу OpenCL
KernelCreate	Создает точку входа в программу OpenCL по указанному индексу
KernelFree	Удаляет функцию запуска OpenCL по указанному индексу

© 2000-2025, MetaQuotes Ltd.

Имя	Описание
SetArgument	Выставляет параметр для функции OpenCL по указанному индексу
SetArgumentBuffer	Выставляет буфер OpenCL в качестве параметра функции OpenCL по указанному индексу
SetArgumentLocalMemory	Выставляет параметр в локальной памяти для функции OpenCL по указанному индексу
SetBuffersCount	Устанавливает количество буферов
SetKernelsCount	Устанавливает количество объектов кернел
Shutdown	Выгружает программу OpenCL
SupportDouble	Узнает наличие поддержки на устройстве вещественных типов данных

© 2000-2025, MetaQuotes Ltd.

BufferCreate

Создает буфер OpenCL по указанному индексу.

```
bool  BufferCreate(
   const int   buffer_index,      // индекс буфера
   const uint  size_in_bytes,     // размер буфера в байтах
   const uint  flags              // комбинация флагов, задающих свойства буфера
   );
```

Параметры

buffer_index

[in] Индекс буфера.

size_in_bytes

[in] Размер буфера в байтах.

flags

[in] Свойства буфера, задаваемые через комбинацию флагов.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

BufferFree

Удаляет буфер по указанному индексу.

```
bool  BufferFree(
   const int  buffer_index      // индекс буфера
   );
```

Параметры

buffer_index

[in] Индекс буфера.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

BufferFromArray

Создает буфер по указанному индексу из массива значений.

```
template<typename T>
bool  BufferFromArray(
   const int   buffer_index,          // индекс буфера
   T           &data[],               // массив значений
   const uint  data_array_offset,     // смещение в массиве значений в байтах
   const uint  data_array_count,      // количество значений из массива для записи
   const uint  flags                  // комбинация флагов, задающих свойства буфера
   );
```

Параметры

buffer_index

[in] Индекс буфера.

&data[]

[in] Массив значений, которые необходимо записать в буфер OpenCL.

data_array_offset

[in] Смещение в массиве значений в байтах, с которого начинается запись.

data_array_count

[in] Количество значений, которые нужно записать.

flags

[in] Свойства буфера, задаваемые через комбинацию флагов.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

BufferRead

Считывает в массив буфер OpenCL по указанному индексу.

```
template<typename T>
bool  BufferRead(
   const int   buffer_index,          // индекс буфера
   T           &data[],               // массив значений
   const uint  cl_buffer_offset,      // смещение в OpenCL буфере в байтах
   const uint  data_array_offset,     // смещение в массиве в элементах
   const uint  data_array_count       // количество значений из буфера для чтения
   );
```

Параметры

buffer_index

[in] Индекс буфера.

&data[]

[in] Массив для получения значений из буфера OpenCL.

cl_buffer_offset

[in] Смещение в OpenCL буфере в байтах, с которого начинается чтение.

data_array_offset

[in] Индекс первого элемента массива для записи значений буфера OpenCL.

data_array_count

[in] Количество значений, которые нужно считать.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

BufferWrite

Записывает массив данных в буфер по указанному индексу.

```
template<typename T>
bool  BufferWrite(
   const int   buffer_index,          // индекс буфера
   T           &data[],               // массив значений
   const uint  cl_buffer_offset,      // смещение в OpenCL буфере в байтах
   const uint  data_array_offset,     // смещение в массиве в элементах
   const uint  data_array_count       // количество значений из массива для записи
   );
```

Параметры

buffer_index

[in] Индекс буфера.

&data[]

[in] Массив значений, которые необходимо записать в буфер OpenCL.

cl_buffer_offset

[in] Смещение в OpenCL буфере в байтах, с которого начинается запись.

data_array_offset

[in] Индекс первого элемента массива, начиная с которого берутся значения из массива для записи в OpenCL буфер.

data_array_count

[in] Количество значений, которые нужно записать.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

Execute

Выполняет OpenCL программу по указанному индексу.

```
bool  Execute(
   const int   kernel_index,           // индекс кернела
   const int   work_dim,               // размерность пространства задач
   const uint  &work_offset[],         // начальное смещение в пространстве задач
   const uint  &work_size[]            // общее количество задач
   );
```

Выполняет OpenCL программу по указанному индексу с заданным количеством задач в локальной группе.

```
bool  Execute(
   const int   kernel_index,           // индекс кернела
   const int   work_dim,               // размерность пространства задач
   const uint  &work_offset[],         // начальное смещение в пространстве задач
   const uint  &work_size[],           // общее количество задач
   const uint  &local_work_size[]      // количество задач в локальной группе
   );
```

Параметры

kernel_index

[in] Индекс объекта кернел.

work_dim

[in] Размерность пространства задач.

&work_offset[]

[in] Начальное смещение в пространстве задач.

&work_size[]

[in] Размер подмножества задач.

&local_work_size[]

[in] Размер локального подмножества задач в группе.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

GetContext

Возвращает хендл контекста OpenCL.

```
int  GetContext();
```

Возвращаемое значение

Хендл контекста OpenCL.

© 2000-2025, MetaQuotes Ltd.

GetKernel

Возвращает хендл объекта кернел по указанному индексу.

```
int  GetKernel(
   const int  kernel_index      // индекс кернела
   );
```

Параметры

kernel_index

[in] Индекс объекта кернел.

Возвращаемое значение

Хендл объекта кернел.

© 2000-2025, MetaQuotes Ltd.

GetKernelName

Возвращает имя объекта кернел по указанному индексу.

```
string  GetKernelName(
   const int  kernel_index      // индекс кернела
   );
```

Параметры

kernel_index

[in] Индекс объекта кернел.

Возвращаемое значение

Имя объекта кернел.

© 2000-2025, MetaQuotes Ltd.

GetProgram

Возвращает хендл программы OpenCL.

```
int  GetProgram();
```

Возвращаемое значение

Хендл программы OpenCL.

© 2000-2025, MetaQuotes Ltd.

Initialize

Инициализирует программу OpenCL.

```
bool  Initialize(
   const string  program,           // хендл программы OpenCL
   const bool    show_log=true      // вести запись в лог
   );
```

Параметры

program

[in] Хендл программы OpenCL.

show_log=true

[in] Включить запись сообщений в журнал.

Возвращаемое значение

Возвращает true, если инициализация прошла успешно. В противном случае возвращает false.

© 2000-2025, MetaQuotes Ltd.

KernelCreate

Создает точку входа в программу OpenCL по указанному индексу.

```
bool  KernelCreate(
   const int     kernel_index,     // индекс кернела
   const string  kernel_name       // имя кернела
   );
```

Параметры

kernel_index

[in] Индекс объекта кернел.

kernel_name

[in] Имя объекта кернел.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

KernelFree

Удаляет функцию запуска OpenCL по указанному индексу.

```
bool  KernelFree(
   const int  kernel_index      // индекс кернела
   );
```

Параметры

kernel_index

[in] Индекс объекта кернел.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

SetArgument

Выставляет параметр для функции OpenCL по указанному индексу.

```
template<typename T>
bool  SetArgument(
   const int  kernel_index,      // индекс кернела
   const int  arg_index,         // индекс аргумента функции
   T          value              // исходный код
   );
```

Параметры

kernel_index

[in] Индекс объекта кернел.

arg_index

[in] Индекс аргумента функции.

value

[in] Значение аргумента функции.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

SetArgumentBuffer

Выставляет буфер OpenCL в качестве параметра функции OpenCL по указанному индексу.

```
bool  SetArgumentBuffer(
   const int  kernel_index,     // индекс кернела
   const int  arg_index,        // индекс аргумента функции
   const int  buffer_index      // индекс буфера
   );
```

Параметры

kernel_index

[in] Индекс объекта кернел.

arg_index

[in] Индекс аргумента функции.

buffer_index

[in] Индекс буфера.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

SetArgumentLocalMemory

Выставляет параметр в локальной памяти для функции OpenCL по указанному индексу.

```
bool  SetArgumentLocalMemory(
   const int  kernel_index,          // индекс кернела
   const int  arg_index,             // индекс аргумента функции
   const int  local_memory_size      // размер локальной памяти
   );
```

Параметры

kernel_index

[in] Индекс объекта кернел.

arg_index

[in] Индекс аргумента функции.

local_memory_size

[in] Размер локальной памяти.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

SetBuffersCount

Устанавливает количество буферов.

```
bool  SetBuffersCount(
   const int  total_buffers      // количество буферов
   );
```

Параметры

total_buffers

[in] Общее количество буферов.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

SetKernelsCount

Устанавливает количество объектов кернел.

```
bool  SetKernelsCount(
   const int  total_kernels      // количество кернелов
   );
```

Параметры

total_kernels

[in] Общее количество кернелов.

Возвращаемое значение

В случае успешного выполнения возвращает true, в противном случае false.

© 2000-2025, MetaQuotes Ltd.

Shutdown

Выгружает программу OpenCL.

```
void  Shutdown();
```

Возвращаемое значение

Нет возвращаемых значений.

© 2000-2025, MetaQuotes Ltd.

SupportDouble

Узнает наличие поддержки на устройстве вещественных типов данных.

```
bool  SupportDouble();
```

Возвращаемое значение

Возвращает true, если на устройстве поддерживаются вещественные типы данных.

© 2000-2025, MetaQuotes Ltd.

Базовый класс CObject

Класс CObject является базовым классом для построения стандартной библиотеки MQL5.

Описание

Класс CObject обеспечивает всем своим потомкам возможность быть элементом связанного списка. Кроме того определяется ряд виртуальных методов для дальнейшей реализации в классах-потомках.

Декларация

```
class CObject
```

Заголовок

```
#include <Object.mqh>
```

Иерархия наследования

CObject

Прямые потомки

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

Методы класса по группам

Атрибуты	
Prev	Получает значение предыдущего элемента
Prev	Устанавливает значение предыдущего элемента
Next	Получает значение последующего элемента
Next	Устанавливает значение последующего элемента
Сравнение	
virtual Compare	Возвращает результат сравнения с другим объектом
Ввод/вывод	
virtual Save	Записывает объект в файл
virtual Load	Читает объект из файла
virtual Type	Возвращает тип объекта

© 2000-2025, MetaQuotes Ltd.

Prev

Получает указатель на предыдущий элемент списка.

```
CObject*  Prev()
```

Возвращаемое значение

Указатель на предыдущий элемент списка. Если элемент в списке первый, то возвращается NULL.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Prev

Устанавливает указатель на предыдущий элемент списка.

```
void  Prev(
   CObject*  object      // указатель на предыдущий элемент списка
   )
```

Параметры

object

[in] Новое значение указателя на предыдущий элемент списка.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Next

Получает указатель на следующий элемент списка.

```
CObject*  Next()
```

Возвращаемое значение

Указатель на следующий элемент списка. Если элемент в списке последний, то возвращается NULL.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Next

Устанавливает указатель на следующий элемент списка.

```
void  Next(
   CObject*  object      // указатель на следующий элемент списка
   )
```

Параметры

object

[in] Новое значение указателя на следующий элемент списка.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Compare

Сравнивает данные элемента списка с данными другого элемента списка.

```
virtual int  Compare(
   const CObject*  node,        // элемент
   const int       mode=0       // вариант
   ) const
```

Параметры

node

[in] Указатель на элемент списка для сравнения

mode=0

[in] Вариант сравнения

Возвращаемое значение

0 - в случае элементы списка равны, -1 - в случае если элемент списка меньше чем элемент списка для сравнения (node), 1 - в случае если элемент списка больше чем элемент списка для сравнения (node).

Примечание

Метод Compare() в классе CObject всегда возвращает 0 и не выполняет каких-либо действий. При необходимости сравнения данных класса-наследника, метод Compare(...) должен быть реализован. Параметр mode нужно использовать при реализации многовариантного сравнения.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- set interconnect
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- compare objects
   int result=object_first.Compare(object_second);
   //--- delete objects
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные элемента списка в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого при помощи функции FileOpen() бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Примечание

Метод Save(int) в классе CObject всегда возвращает true и не выполняет каких-либо действий. При необходимости сохранения данных класса-наследника в файле, метод Save(int) должен быть реализован.

Пример:

```
//--- 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);
```

© 2000-2025, MetaQuotes Ltd.

```
         //---
         return;
        }
      FileClose(file_handle);
     }
   delete object;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные элемента списка из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого при помощи функции FileOpen() бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Примечание

Метод Load(int) в классе CObject всегда возвращает true и не выполняет каких-либо действий. При необходимости загрузки данных класса-наследника из файла, метод Load(int) должен быть реализован.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       FileClose(file_handle);
      }
   //--- use object
   //--- . . .
   delete object;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа (для CObject - 0).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Структуры данных

Этот раздел содержит технические детали работы с различными структурами данных (массивами, связанными списками и др.) и описание соответствующих компонентов стандартной библиотеки MQL5.

Использование классов структур данных позволит сэкономить время при создании пользовательских хранилищ данных разнообразных форматов (в том числе составных структур данных).

Стандартная библиотека MQL5 (в части наборов данных) размещается в рабочем каталоге терминала в папке Include\Arrays

Массивы данных

Использование классов динамических массивов данных позволит сэкономить время при создании пользовательских хранилищ данных разнообразных форматов (в том числе многомерных массивов).

Стандартная библиотека MQL5 (в части массивов данных) размещается в рабочем каталоге терминала в папке Include\Arrays.

Класс	Описание
CArray	Базовый класс динамического массива данных
CArrayChar	Динамический массив переменных типа char или uchar
CArrayShort	Динамический массив переменных типа short или ushort
CArrayInt	Динамический массив переменных типа int или uint
CArrayLong	Динамический массив переменных типа long или ulong
CArrayFloat	Динамический массив переменных типа float
CArrayDouble	Динамический массив переменных типа double
CArrayString	Динамический массив переменных типа string
CArrayObj	Динамический массив указателей CObject
CList	Обеспечивает возможность работы со списком экземпляров класса CObject и его наследников

© 2000-2025, MetaQuotes Ltd.

Класс	Описание
CTreeNode	Обеспечивает возможность работы с узлами двоичного дерева CTree
CTree	Обеспечивает возможность работы с двоичным деревом экземпляров класса CTreeNode и его наследников.

© 2000-2025, MetaQuotes Ltd.

CArray

Класс CArray является базовым классом динамического массива переменных.

Описание

Класс CArray предназначен для работы с динамическим массивом переменных: распределение памяти, сортировка и работа с файлами.

Декларация

```
class CArray : public CObject
```

Заголовок

```
#include <Arrays\Array.mqh>
```

Иерархия наследования

CObject

CArray

Прямые потомки

CArrayChar, CArrayDouble, CArrayFloat, CArrayInt, CArrayLong, CArrayObj, CArrayShort, CArrayString

Методы класса по группам

Атрибуты	
Step	Получает шаг приращения размеров массива
Step	Устанавливает шаг приращения размеров массива
Total	Получает количество элементов в массиве
Available	Получает количество свободных элементов массива, доступных без дополнительного распределения памяти
Max	Получает максимально возможный размер массива без перевыделения памяти
IsSorted	Получает признак сортированности массива по указанному варианту
SortMode	Получает вариант сортировки массива
Метод очистки	
Clear	Удаляет все элементы массива без освобождения памяти
Сортировка	
Sort	Сортирует массив по указанному варианту
Ввод/вывод	

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
virtual Save	Сохраняет данные массива в файле
virtual Load	Загружает данные массива из файла

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Step

Получает шаг приращения размеров массива.

```
int  Step() const
```

Возвращаемое значение

Шаг приращения размеров массива.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Step

Устанавливает шаг приращения размеров массива.

```
bool  Step(
   int  step      // шаг
   )
```

Параметры

step

[in] Новое значение шага приращения размеров массива.

Возвращаемое значение

true в случае успеха, false - если была попытка установить шаг меньше или равный нулю.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Total

Получает количество элементов в массиве.

```
int  Total() const;
```

Возвращаемое значение

Количество элементов в массиве.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Available

Получает количество свободных элементов массива, доступных без дополнительного распределения памяти.

```
int  Available() const
```

Возвращаемое значение

Количество свободных элементов массива, доступных без дополнительного распределения памяти.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Max

Получает максимально возможный размер массива без перевыделения памяти.

```
int  Max() const
```

Возвращаемое значение

Максимально возможный размер массива без перевыделения памяти.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

IsSorted

Получает признак сортированности массива по указанному варианту.

```
bool  IsSorted(
   int  mode=0       // вариант сортировки
   ) const
```

Параметры

mode=0

[in] Проверяемый вариант сортировки.

Возвращаемое значение

Флаг сортированности списка. Если список сортированный по указанному варианту — true, иначе — false.

Примечание

Признак сортированности массива нельзя изменить непосредственно. Признак сортированности устанавливается методом Sort() и сбрасывается любыми методами добавления/вставки кроме InsertSort(...).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SortMode

Получает вариант сортировки массива

```
int  SortMode() const;
```

Возвращаемое значение

Вариант сортировки

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы массива без освобождения памяти.

```
void  Clear()
```

Возвращаемое значение

Нет.

Пример:

```
//--- 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;
 }
```

© 2000-2025, MetaQuotes Ltd.

Sort

Сортирует массив по указанному варианту.

```
void  Sort(
   int  mode=0       // вариант сортировки
   )
```

Параметры

mode=0

[in] Вариант сортировки массива.

Возвращаемое значение

Нет.

Примечание

Сортировка массива производится всегда по возрастанию. Для массивов простых типов данных (CArrayChar, CArrayShort и др.) параметр mode не используется. Для массива CArrayObj мультивариантные сортировки должны быть реализованы в методе Sort(int) классов-наследников.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные массива в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого при помощи функции FileOpen(...) бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные массива из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого при помощи функции FileOpen(...) бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

CArrayChar

Класс CArrayChar является классом динамического массива переменных типа char или uchar.

Описание

Класс CArrayChar обеспечивает возможность работы с динамическим массивом переменных типа char или uchar. В классе реализованы возможности добавления/вставки/удаления элементов массива, сортировки массива, поисков в сортированном массиве. Кроме того, реализованы методы работы с файлом.

Декларация

```
class CArrayChar : public CArray
```

Заголовок

```
#include <Arrays\ArrayChar.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayChar

Методы класса по группам

Управление памятью	
Reserve	Резервирует память для увеличения размеров массива
Resize	Устанавливает новый (меньший) размер массива
Shutdown	Очищает массив с полным освобождением памяти
Наполнение	
Add	Добавляет элемент в конец массива
AddArray	Добавляет в конец массива элементы из другого массива
AddArray	Добавляет в конец массива элементы из другого массива
Insert	Вставляет элемент в массив в указанную позицию
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
AssignArray	Копирует в массив элементы из другого массива

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
AssignArray	Копирует в массив элементы из другого массива
Изменение	
Update	Изменяет элемент в указанной позиции массива
Shift	Перемещает элемент из указанной позиции массива на указанное смещение
Удаление	
Delete	Удаляет элемент из указанной позиции массива
DeleteRange	Удаляет группу элементов из указанной позиции массива
Доступ	
At	Получает элемент из указанной позиции массива
Сравнение	
CompareArray	Сравнивает массив с другим массивом
CompareArray	Сравнивает массив с другим массивом
Операции с сортированным массивом	
InsertSort	Вставляет элемент в сортированный массив
Search	Ищет элемент равный образцу в сортированном массиве
SearchGreat	Ищет элемент больше образца в сортированном массиве
SearchLess	Ищет элемент меньше образца в сортированном массиве
SearchGreatOrEqual	Ищет элемент больше или равный образцу в сортированном массиве
SearchLessOrEqual	Ищет элемент меньше или равный образцу в сортированном массиве
SearchFirst	Ищет первый элемент равный образцу в сортированном массиве
SearchLast	Ищет последний элемент равный образцу в сортированном массиве
SearchLinear	Ищет элемент равный образцу в массиве
Ввод/вывод	
virtual Save	Сохраняет данные массива в файле
virtual Load	Загружает данные массива из файла

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
virtual Type	Получает идентификатор типа массива

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Резервирует память для увеличения размеров массива.

```
bool  Reserve(
   int  size      // количество
   )
```

Параметры

size

[in] Количество дополнительных элементов массива.

Возвращаемое значение

true в случае успеха, false - если была попытка запросить количество меньше или равное нулю, либо если массив увеличить не удалось.

Примечание

Для уменьшения фрагментации памяти, приращение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Устанавливает новый (меньший) размер массива.

```
bool  Resize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер массива.

Возвращаемое значение

true в случае успеха, false - если была попытка установить размер меньше нуля.

Примечание

Изменение размера массива позволяет оптимально использовать память. Лишние элементы справа теряются. Для уменьшения фрагментации памяти, изменение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Очищает массив с полным освобождением памяти.

```
bool  Shutdown()
```

Возвращаемое значение

true в случае успеха, false - если произошла ошибка.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в конец массива.

```
bool  Add(
   char  element      // элемент для добавления
   )
```

Параметры

element

[in] Значение элемента для добавления в массив.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const char&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const CArrayChar*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayChar-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в массив в указанную позицию.

```
bool  Insert(
   char  element,     // элемент для вставки
   int   pos          // позиция
   )
```

Параметры

element

[in] Значение элемента для вставки в массив

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанной позиции.

```
bool  InsertArray(
   const char&  src[],     // массив-источник
   int           pos       // позиция
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для вставки

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанной позиции.

```
bool  InsertArray(
   CArrayChar*  src,      // указатель на источник
   int           pos      // позиция
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayChar-источник элементов для вставки.

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
      delete array;
      return;
     }
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const char&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const CArrayChar*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayChar-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- arrays is identical
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Изменяет элемент в указанной позиции массива.

```
bool  Update(
   int   pos,          // позиция
   char  element       // значение
   )
```

Параметры

pos

[in] Позиция элемента в массиве для изменения

element

[in] Новое значение элемента

Возвращаемое значение

true в случае успеха, false - если нет возможности изменить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Перемещает элемент из указанной позиции массива на указанное смещение.

```
bool  Shift(
   int  pos,        // позиция
   int  shift       // смещение
   )
```

Параметры

pos

[in] Позиция перемещаемого элемента массиве

shift

[in] Значение смещения (как положительное, так и отрицательное значение).

Возвращаемое значение

true в случае успеха, false - если нет возможности переместить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из указанной позиции массива.

```
bool  Delete(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Удаляет группу элементов из указанной позиции массива.

```
bool  DeleteRange(
   int  from,      // позиция первого элемента
   int  to         // позиция последнего элемента
   )
```

Параметры

from

[in] Позиция первого удаляемого элемента в массиве.

to

[in] Позиция последнего удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент из указанной позиции массива.

```
char  At(
   int  pos      // позиция
   ) const
```

Параметры

pos

[in] Позиция искомого элемента в массиве.

Возвращаемое значение

Значение элемента в случае успеха, CHAR_MAX- если была попытка получить элемент из не существующей позиции (при этом код последней ошибки ERR_OUT_OF_RANGE).

Примечание

Разумеется, CHAR_MAX может быть и валидным значением элемента массива, поэтому, получив такое значение, всегда проверяйте код последней ошибки.

Пример:

```
//--- 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)
        {
         //--- ошибка чтения из массива
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- . . .
     }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const char&  src[]      // массив-источник
   ) const
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const CArrayChar*  src      // указатель на источник
   ) const
```

Параметры

src

[in] Указатель на экземпляр класса CArrayChar-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Вставляет элемент в сортированный массив.

```
bool  InsertSort(
   char  element      // элемент для вставки
   )
```

Параметры

element

[in] Значение элемента для вставки в сортированный массив

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

Ищет элемент равный образцу в сортированном массиве.

```
int  Search(
   char  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Ищет элемент больше образца в сортированном массиве.

```
int  SearchGreat(
   char  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Ищет элемент меньше образца в сортированном массиве.

```
int  SearchLess(
   char  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Ищет элемент больше или равный образцу в сортированном массиве.

```
int  SearchGreatOrEqual(
   char  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Ищет элемент меньше или равный образцу в сортированном массиве.

```
int  SearchLessOrEqual(
   char  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Ищет первый элемент равный образцу в сортированном массиве.

```
int  SearchFirst(
   char  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Ищет последний элемент равный образцу в сортированном массиве.

```
int  SearchLast(
   char  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Ищет элемент равный образцу в массиве.

```
int  SearchLinear(
   char  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Примечание

Поиск производится последовательным перебором элементов массива (линейный поиск для несортированных массивов).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные массива в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные массива из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       printf("Element[%d] = '%c'",i,array.At(i));
      }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа массива.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа массива (для CArrayChar - 77).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayShort

Класс CArrayShort является классом динамического массива переменных типа short или ushort.

Описание

Класс CArrayShort обеспечивает возможность работы с динамическим массивом переменных типа short или ushort. В классе реализованы возможности добавления/вставки/удаления элементов массива, сортировки массива, поисков в сортированном массиве. Кроме того, реализованы методы работы с файлом.

Декларация

```
class CArrayShort : public CArray
```

Заголовок

```
#include <Arrays\ArrayShort.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayShort

Методы класса по группам

Управление памятью	
Reserve	Резервирует память для увеличения размеров массива
Resize	Устанавливает новый (меньший) размер массива
Shutdown	Очищает массив с полным освобождением памяти
Наполнение	
Add	Добавляет элемент в конец массива
AddArray	Добавляет в конец массива элементы из другого массива
AddArray	Добавляет в конец массива элементы из другого массива
Insert	Вставляет элемент в массив в указанную позицию
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
AssignArray	Копирует в массив элементы из другого массива

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
AssignArray	Копирует в массив элементы из другого массива
Изменение	
Update	Изменяет элемент в указанной позиции массива
Shift	Перемещает элемент из указанной позиции массива на указанное смещение
Удаление	
Delete	Удаляет элемент из указанной позиции массива
DeleteRange	Удаляет группу элементов из указанной позиции массива
Доступ	
At	Получает элемент из указанной позиции массива
Сравнение	
CompareArray	Сравнивает массив с другим массивом
CompareArray	Сравнивает массив с другим массивом
Операции с сортированным массивом	
InsertSort	Вставляет элемент в сортированный массив
Search	Ищет элемент равный образцу в сортированном массиве
SearchGreat	Ищет элемент больше образца в сортированном массиве
SearchLess	Ищет элемент меньше образца в сортированном массиве
SearchGreatOrEqual	Ищет элемент больше или равный образцу в сортированном массиве
SearchLessOrEqual	Ищет элемент меньше или равный образцу в сортированном массиве
SearchFirst	Ищет первый элемент равный образцу в сортированном массиве
SearchLast	Ищет последний элемент равный образцу в сортированном массиве
SearchLinear	Ищет элемент равный образцу в массиве
Ввод/вывод	
virtual Save	Сохраняет данные массива в файле
virtual Load	Загружает данные массива из файла

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
virtual Type	Получает идентификатор типа массива

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Резервирует память для увеличения размеров массива.

```
bool  Reserve(
   int  size      // количество
   )
```

Параметры

size

[in] Количество дополнительных элементов массива.

Возвращаемое значение

true в случае успеха, false - если была попытка запросить количество меньше или равное нулю, либо если массив увеличить не удалось.

Примечание

Для уменьшения фрагментации памяти, приращение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Устанавливает новый (меньший) размер массива.

```
bool  Resize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер массива.

Возвращаемое значение

true в случае успеха, false - если была попытка установить размер меньше нуля.

Примечание

Изменение размера массива позволяет оптимально использовать память. Лишние элементы справа теряются. Для уменьшения фрагментации памяти, изменение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Очищает массив с полным освобождением памяти.

```
bool  Shutdown()
```

Возвращаемое значение

true в случае успеха, false - если произошла ошибка.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в конец массива.

```
bool  Add(
   short  element      // элемент для добавления
   )
```

Параметры

element

[in] Значение элемента для добавления в массив.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const short&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const CArrayShort*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayShort-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в массив в указанную позицию.

```
bool  Insert(
   short  element,      // элемент для вставки
   int    pos           // позиция
   )
```

Параметры

element

[in] Значение элемента для вставки в массив

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   const short&  src[],      // массив-источник
   int            pos         // позиция
   )
```

Параметры

src[]

[in] ссылка на массив-источник элементов для вставки

pos

[in] позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   CArrayShort*  src,      // указатель на источник
   int            pos      // позиция
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayShort-источник элементов для вставки.

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
      delete array;
      return;
     }
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const short&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const CArrayShort*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayShort-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- arrays is identical
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Изменяет элемент в указанной позиции массива.

```
bool  Update(
   int    pos,          // позиция
   short  element       // значение
   )
```

Параметры

pos

[in] Позиция элемента в массиве для изменения

element

[in] Новое значение элемента

Возвращаемое значение

true в случае успеха, false - если нет возможности изменить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Перемещает элемент из указанной позиции массива на указанное смещение.

```
bool  Shift(
   int  pos,        // позиция
   int  shift       // смещение
   )
```

Параметры

pos

[in] Позиция перемещаемого элемента массиве

shift

[in] Значение смещения (как положительное, так и отрицательное значение).

Возвращаемое значение

true в случае успеха, false - если нет возможности переместить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из указанной позиции массива.

```
bool  Delete(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Удаляет группу элементов из указанной позиции массива.

```
bool  DeleteRange(
   int  from,      // позиция первого элемента
   int  to         // позиция последнего элемента
   )
```

Параметры

from

[in] Позиция первого удаляемого элемента в массиве.

to

[in] Позиция последнего удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент из указанной позиции массива.

```
short  At(
   int  pos      // позиция
   ) const
```

Параметры

pos

[in] Позиция искомого элемента в массиве.

Возвращаемое значение

Значение элемента в случае успеха, SHORT_MAX- если была попытка получить элемент из не существующей позиции (при этом код последней ошибки ERR_OUT_OF_RANGE).

Примечание

Разумеется, SHORT_MAX может быть и валидным значением элемента массива, поэтому, получив такое значение, всегда проверяйте код последней ошибки.

Пример:

```
//--- 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)
        {
         //--- ошибка чтения из массива
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- . . .
     }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const short&  src[]      // массив-источник
   ) const
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const CArrayShort*  src      // указатель на источник
   ) const
```

Параметры

src

[in] Указатель на экземпляр класса CArrayShort-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Вставляет элемент в сортированный массив.

```
bool  InsertSort(
   short  element      // элемент для вставки
   )
```

Параметры

element

[in] Значение элемента для вставки в сортированный массив

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

Ищет элемент равный образцу в сортированном массиве.

```
int  Search(
   short  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Ищет элемент больше образца в сортированном массиве.

```
int  SearchGreat(
   short  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Ищет элемент меньше образца в сортированном массиве.

```
int  SearchLess(
   short  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Ищет элемент больше или равный образцу в сортированном массиве.

```
int  SearchGreatOrEqual(
   short  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Ищет элемент меньше или равный образцу в сортированном массиве.

```
int  SearchLessOrEqual(
   short  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Ищет первый элемент равный образцу в сортированном массиве.

```
int  SearchFirst(
   short  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Ищет последний элемент равный образцу в сортированном массиве.

```
int  SearchLast(
   short  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Ищет элемент, равный образцу в массиве.

```
int  SearchLinear(
   short  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Примечание

Поиск производится последовательным перебором элементов массива (линейный поиск для несортированных массивов).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные массива в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       FileClose(file_handle);
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные массива из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       printf("Element[%d] = %d",i,array.At(i));
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа массива.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа массива (для CArrayShort - 82).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayInt

Класс CArrayInt является классом динамического массива переменных типа int или uint.

Описание

Класс CArrayInt обеспечивает возможность работы с динамическим массивом переменных типа int или uint. В классе реализованы возможности добавления/вставки/удаления элементов массива, сортировки массива, поисков в сортированном массиве. Кроме того, реализованы методы работы с файлом.

Декларация

```
class CArrayInt : public CArray
```

Заголовок

```
#include <Arrays\ArrayInt.mqh>
```

Иерархия наследования

CObject

CArray

CArrayInt

Прямые потомки

CSpreadBuffer

Методы класса по группам

Управление памятью	
Reserve	Резервирует память для увеличения размеров массива
Resize	Устанавливает новый (меньший) размер массива
Shutdown	Очищает массив с полным освобождением памяти
Наполнение	
Add	Добавляет элемент в конец массива
AddArray	Добавляет в конец массива элементы из другого массива
AddArray	Добавляет в конец массива элементы из другого массива
Insert	Вставляет элемент в массив в указанную позицию
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
AssignArray	Копирует в массив элементы из другого массива
AssignArray	Копирует в массив элементы из другого массива
Изменение	
Update	Изменяет элемент в указанной позиции массива
Shift	Перемещает элемент из указанной позиции массива на указанное смещение
Удаление	
Delete	Удаляет элемент из указанной позиции массива
DeleteRange	Удаляет группу элементов из указанной позиции массива
Доступ	
At	Получает элемент из указанной позиции массива
Сравнение	
CompareArray	Сравнивает массив с другим массивом
CompareArray	Сравнивает массив с другим массивом
Операции с сортированным массивом	
InsertSort	Вставляет элемент в сортированный массив
Search	Ищет элемент равный образцу в сортированном массиве
SearchGreat	Ищет элемент больше образца в сортированном массиве
SearchLess	Ищет элемент меньше образца в сортированном массиве
SearchGreatOrEqual	Ищет элемент больше или равный образцу в сортированном массиве
SearchLessOrEqual	Ищет элемент меньше или равный образцу в сортированном массиве
SearchFirst	Ищет первый элемент равный образцу в сортированном массиве
SearchLast	Ищет последний элемент равный образцу в сортированном массиве
SearchLinear	Ищет элемент равный образцу в массиве

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
Ввод/вывод	
virtual Save	Сохраняет данные массива в файле
virtual Load	Загружает данные массива из файла
virtual Type	Получает идентификатор типа массива

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Резервирует память для увеличения размеров массива.

```
bool  Reserve(
   int  size      // количество
   )
```

Параметры

size

[in] Количество дополнительных элементов массива.

Возвращаемое значение

true в случае успеха, false - если была попытка запросить количество меньше или равное нулю, либо если массив увеличить не удалось.

Примечание

Для уменьшения фрагментации памяти, приращение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Устанавливает новый (меньший) размер массива.

```
bool  Resize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер массива.

Возвращаемое значение

true в случае успеха, false - если была попытка установить размер меньше нуля.

Примечание

Изменение размера массива позволяет оптимально использовать память. Лишние элементы справа теряются. Для уменьшения фрагментации памяти, изменение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Очищает массив с полным освобождением памяти.

```
bool  Shutdown()
```

Возвращаемое значение

true в случае успеха, false - если произошла ошибка.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в конец массива.

```
bool  Add(
   int  element      // элемент для добавления
   )
```

Параметры

element

[in] Значение элемента для добавления в массив.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const int&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const CArrayInt*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayInt-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в массив в указанную позицию.

```
bool  Insert(
   int  element,     // элемент для вставки
   int  pos          // позиция
   )
```

Параметры

element

[in] Значение элемента для вставки в массив

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   const int&  src[],     // массив-источник
   int          pos        // позиция
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для вставки

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   CArrayInt*  src,      // указатель на источник
   int          pos      // позиция
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayInt-источник элементов для вставки.

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
      delete array;
      return;
     }
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const int&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const CArrayInt*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayInt-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- arrays is identical
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Изменяет элемент в указанной позиции массива.

```
bool  Update(
   int  pos,          // позиция
   int  element       // значение
   )
```

Параметры

pos

[in] Позиция элемента в массиве для изменения

element

[in] Новое значение элемента

Возвращаемое значение

true в случае успеха, false - если нет возможности изменить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Перемещает элемент из указанной позиции массива на указанное смещение.

```
bool  Shift(
   int  pos,        // позиция
   int  shift       // смещение
   )
```

Параметры

pos

[in] Позиция перемещаемого элемента массиве

shift

[in] Значение смещения (как положительное, так и отрицательное значение).

Возвращаемое значение

true в случае успеха, false - если нет возможности переместить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из указанной позиции массива.

```
bool  Delete(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Удаляет группу элементов из указанной позиции массива.

```
bool  DeleteRange(
   int  from,      // позиция первого элемента
   int  to         // позиция последнего элемента
   )
```

Параметры

from

[in] Позиция первого удаляемого элемента в массиве.

to

[in] Позиция последнего удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент из указанной позиции массива.

```
int  At(
   int  pos      // позиция
   ) const
```

Параметры

pos

[in] Позиция искомого элемента в массиве.

Возвращаемое значение

Значение элемента в случае успеха, INT_MAX- если была попытка получить элемент из не существующей позиции (при этом код последней ошибки ERR_OUT_OF_RANGE).

Примечание

Разумеется, INT_MAX может быть и валидным значением элемента массива, поэтому, получив такое значение, всегда проверяйте код последней ошибки.

Пример:

```
//--- 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)
        {
         //--- ошибка чтения из массива
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- . . .
     }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const int&  src[]      // массив-источник
   ) const
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const CArrayInt*  src      // указатель на источник
   ) const
```

Параметры

src

[in] Указатель на экземпляр класса CArrayInt-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Вставляет элемент в сортированный массив.

```
bool  InsertSort(
   int  element      // элемент для вставки
   )
```

Параметры

element

[in] Значение элемента для вставки в сортированный массив

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

Ищет элемент равный образцу в сортированном массиве.

```
int  Search(
   int  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Ищет элемент больше образца в сортированном массиве.

```
int  SearchGreat(
   int  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Ищет элемент меньше образца в сортированном массиве.

```
int  SearchLess(
   int  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Ищет элемент больше или равный образцу в сортированном массиве.

```
int  SearchGreatOrEqual(
   int  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Ищет элемент меньше или равный образцу в сортированном массиве.

```
int  SearchLessOrEqual(
   int  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Ищет первый элемент равный образцу в сортированном массиве.

```
int  SearchFirst(
   int  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Ищет последний элемент равный образцу в сортированном массиве.

```
int  SearchLast(
   int  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Ищет элемент равный образцу в массиве.

```
int  SearchLinear(
   int  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Примечание

Поиск производится последовательным перебором элементов массива (линейный поиск для несортированных массивов).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные массива в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       FileClose(file_handle);
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные массива из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       printf("Element[%d] = %d",i,array.At(i));
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа массива.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа массива (для CArrayInt - 82).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayLong

Класс CArrayLong является классом динамического массива переменных типа long или ulong.

Описание

Класс CArrayLong обеспечивает возможность работы с динамическим массивом переменных типа long или ulong. В классе реализованы возможности добавления/вставки/удаления элементов массива, сортировки массива, поисков в сортированном массиве. Кроме того, реализованы методы работы с файлом.

Декларация

```
class CArrayLong : public CArray
```

Заголовок

```
#include <Arrays\ArrayLong.mqh>
```

Иерархия наследования

CObject

CArray

CArrayLong

Прямые потомки

CRealVolumeBuffer, CTickVolumeBuffer, CTimeBuffer

Методы класса по группам

Управление памятью	
Reserve	Резервирует память для увеличения размеров массива
Resize	Устанавливает новый (меньший) размер массива
Shutdown	Очищает массив с полным освобождением памяти
Наполнение	
Add	Добавляет элемент в конец массива
AddArray	Добавляет в конец массива элементы из другого массива
AddArray	Добавляет в конец массива элементы из другого массива
Insert	Вставляет элемент в массив в указанную позицию
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
AssignArray	Копирует в массив элементы из другого массива
AssignArray	Копирует в массив элементы из другого массива
Изменение	
Update	Изменяет элемент в указанной позиции массива
Shift	Перемещает элемент из указанной позиции массива на указанное смещение
Удаление	
Delete	Удаляет элемент из указанной позиции массива
DeleteRange	Удаляет группу элементов из указанной позиции массива
Доступ	
At	Получает элемент из указанной позиции массива
Сравнение	
CompareArray	Сравнивает массив с другим массивом
CompareArray	Сравнивает массив с другим массивом
Операции с сортированным массивом	
InsertSort	Вставляет элемент в сортированный массив
Search	Ищет элемент равный образцу в сортированном массиве
SearchGreat	Ищет элемент больше образца в сортированном массиве
SearchLess	Ищет элемент меньше образца в сортированном массиве
SearchGreatOrEqual	Ищет элемент больше или равный образцу в сортированном массиве
SearchLessOrEqual	Ищет элемент меньше или равный образцу в сортированном массиве
SearchFirst	Ищет первый элемент равный образцу в сортированном массиве
SearchLast	Ищет последний элемент равный образцу в сортированном массиве
SearchLinear	Ищет элемент равный образцу в массиве

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
Ввод/вывод	
virtual Save	Сохраняет данные массива в файле
virtual Load	Загружает данные массива из файла
virtual Type	Получает идентификатор типа массива

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Резервирует память для увеличения размеров массива.

```
bool  Reserve(
   int  size      // количество
   )
```

Параметры

size

[in] Количество дополнительных элементов массива.

Возвращаемое значение

true в случае успеха, false - если была попытка запросить количество меньше или равное нулю, либо если массив увеличить не удалось.

Примечание

Для уменьшения фрагментации памяти, приращение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Устанавливает новый (меньший) размер массива.

```
bool  Resize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер массива.

Возвращаемое значение

true в случае успеха, false - если была попытка установить размер меньше нуля.

Примечание

Изменение размера массива позволяет оптимально использовать память. Лишние элементы справа теряются. Для уменьшения фрагментации памяти, изменение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Очищает массив с полным освобождением памяти.

```
bool  Shutdown()
```

Возвращаемое значение

true в случае успеха, false - если произошла ошибка.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в конец массива.

```
bool  Add(
   long  element      // элемент для добавления
   )
```

Параметры

element

[in] Значение элемента для добавления в массив.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const long&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const CArrayLong*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayLong-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в массив в указанную позицию.

```
bool  Insert(
   long  element,     // элемент для вставки
   int   pos          // позиция
   )
```

Параметры

element

[in] Значение элемента для вставки в массив

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   const long&  src[],     // массив-источник
   int           pos        // позиция
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для вставки

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   CArrayLong*  src,      // указатель на источник
   int           pos      // позиция
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayLong-источник элементов для вставки.

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
      delete array;
      return;
     }
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const long&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const CArrayLong*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayLong-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- arrays is identical
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Изменяет элемент в указанной позиции массива.

```
bool  Update(
   int   pos,          // позиция
   long  element       // значение
   )
```

Параметры

pos

[in] Позиция элемента в массиве для изменения

element

[in] Новое значение элемента

Возвращаемое значение

true в случае успеха, false - если нет возможности изменить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Перемещает элемент из указанной позиции массива на указанное смещение.

```
bool  Shift(
   int  pos,        // позиция
   int  shift       // смещение
   )
```

Параметры

pos

[in] Позиция перемещаемого элемента массиве

shift

[in] Значение смещения (как положительное, так и отрицательное значение).

Возвращаемое значение

true в случае успеха, false - если нет возможности переместить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из указанной позиции массива.

```
bool  Delete(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Удаляет группу элементов из указанной позиции массива.

```
bool  DeleteRange(
   int  from,      // позиция первого элемента
   int  to         // позиция последнего элемента
   )
```

Параметры

from

[in] Позиция первого удаляемого элемента в массиве.

to

[in] Позиция последнего удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент из указанной позиции массива.

```
long  At(
   int  pos      // позиция
   ) const
```

Параметры

pos

[in] Позиция искомого элемента в массиве.

Возвращаемое значение

Значение элемента в случае успеха, LONG_MAX если была попытка получить элемент из не существующей позиции (при этом код последней ошибки ERR_OUT_OF_RANGE).

Примечание

Разумеется, LONG_MAX может быть и валидным значением элемента массива, поэтому, получив такое значение, всегда проверяйте код последней ошибки.

Пример:

```
//--- 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)
        {
         //--- ошибка чтения из массива
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- . . .
     }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const long&  src[]      // массив-источник
   ) const
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArrayconst

Сравнивает массив с другим массивом.

```
bool  CompareArrayconst(
   const CArrayLong*  src      // указатель на источник
   ) const
```

Параметры

src

[in] Указатель на экземпляр класса CArrayLong-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Вставляет элемент в сортированный массив.

```
bool  InsertSort(
   long  element      // элемент для вставки
   )
```

Параметры

element

[in] Значение элемента для вставки в сортированный массив

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

Ищет элемент равный образцу в сортированном массиве.

```
int  Search(
   long  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Ищет элемент больше образца в сортированном массиве.

```
int  SearchGreat(
   long  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Ищет элемент меньше образца в сортированном массиве.

```
int  SearchLess(
   long  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Ищет элемент больше или равный образцу в сортированном массиве.

```
int  SearchGreatOrEqual(
   long  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Ищет элемент меньше или равный образцу в сортированном массиве.

```
int  SearchLessOrEqual(
   long  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Ищет первый элемент равный образцу в сортированном массиве.

```
int  SearchFirst(
   long  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Ищет последний элемент равный образцу в сортированном массиве.

```
int  SearchLast(
   long  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Ищет элемент равный образцу в массиве.

```
int  SearchLinear(
   long  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Примечание

Поиск производится последовательным перебором элементов массива (линейный поиск для несортированных массивов).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные массива в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       FileClose(file_handle);
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные массива из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       printf("Element[%d] = %I64",i,array.At(i));
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа массива.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа массива (для CArrayLong - 84).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayFloat

Класс CArrayFloat является классом динамического массива переменных типа float.

Описание

Класс CArrayFloat обеспечивает возможность работы с динамическим массивом переменных типа float. В классе реализованы возможности добавления/вставки/удаления элементов массива, сортировки массива, поисков в сортированном массиве. Кроме того, реализованы методы работы с файлом.

Декларация

```
class CArrayFloat : public CArray
```

Заголовок

```
#include <Arrays\ArrayFloat.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayFloat

Методы класса по группам

Атрибуты	
Delta	Устанавливает величину допуска сравнения
Управление памятью	
Reserve	Резервирует память для увеличения размеров массива
Resize	Устанавливает новый (меньший) размер массива
Shutdown	Очищает массив с полным освобождением памяти
Наполнение	
Add	Добавляет элемент в конец массива
AddArray	Добавляет в конец массива элементы из другого массива
AddArray	Добавляет в конец массива элементы из другого массива
Insert	Вставляет элемент в массив в указанную позицию
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
AssignArray	Копирует в массив элементы из другого массива
AssignArray	Копирует в массив элементы из другого массива
Изменение	
Update	Изменяет элемент в указанной позиции массива
Shift	Перемещает элемент из указанной позиции массива на указанное смещение
Удаление	
Delete	Удаляет элемент из указанной позиции массива
DeleteRange	Удаляет группу элементов из указанной позиции массива
Доступ	
At	Получает элемент из указанной позиции массива
Сравнение	
CompareArray	Сравнивает массив с другим массивом
CompareArray	Сравнивает массив с другим массивом
Операции с сортированным массивом	
InsertSort	Вставляет элемент в сортированный массив
Search	Ищет элемент равный образцу в сортированном массиве
SearchGreat	Ищет элемент больше образца в сортированном массиве
SearchLess	Ищет элемент меньше образца в сортированном массиве
SearchGreatOrEqual	Ищет элемент больше или равный образцу в сортированном массиве
SearchLessOrEqual	Ищет элемент меньше или равный образцу в сортированном массиве
SearchFirst	Ищет первый элемент равный образцу в сортированном массиве
SearchLast	Ищет последний элемент равный образцу в сортированном массиве
SearchLinear	Ищет элемент равный образцу в массиве

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
Ввод/вывод	
virtual Save	Сохраняет данные массива в файле
virtual Load	Загружает данные массива из файла
virtual Type	Получает идентификатор типа массива

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Delta

Устанавливает допуск сравнения.

```
void  Delta(
   float  delta      // допуск
   )
```

Параметры

delta

[in] новое значение допуска сравнения.

Возвращаемое значение

Нет

Примечание

Допуск сравнения используется при поисках. Значения считаются равными, если их разница меньше или равна допуску. По умолчанию допуск равен 0.0.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

Резервирует память для увеличения размеров массива.

```
bool  Reserve(
   int  size      // количество
   )
```

Параметры

size

[in] Количество дополнительных элементов массива.

Возвращаемое значение

true в случае успеха, false - если была попытка запросить количество меньше или равное нулю, либо если массив увеличить не удалось.

Примечание

Для уменьшения фрагментации памяти, приращение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Устанавливает новый (меньший) размер массива.

```
bool  Resize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер массива.

Возвращаемое значение

true в случае успеха, false - если была попытка установить размер меньше нуля.

Примечание

Изменение размера массива позволяет оптимально использовать память. Лишние элементы справа теряются. Для уменьшения фрагментации памяти, изменение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Очищает массив с полным освобождением памяти.

```
bool  Shutdown()
```

Возвращаемое значение

true в случае успеха, false - если произошла ошибка.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в конец массива.

```
bool  Add(
   float  element      // элемент для добавления
   )
```

Параметры

element

[in] Значение элемента для добавления в массив.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const float&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const CArrayFloat*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayFloat-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в массив в указанную позицию.

```
bool  Insert(
   float  element,      // элемент для вставки
   int    pos           // позиция
   )
```

Параметры

element

[in] Значение элемента для вставки в массив

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанной позиции.

```
bool  InsertArray(
   const float&  src[],     // массив-источник
   int           pos        // позиция
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для вставки

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   CArrayFloat*  src,      // указатель на источник
   int            pos      // позиция
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayFloat-источник элементов для вставки.

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
      delete array;
      return;
     }
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const float&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const CArrayFloat*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayFloat-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- arrays is identical
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Изменяет элемент в указанной позиции массива.

```
bool  Update(
   int    pos,          // позиция
   float  element       // значение
   )
```

Параметры

pos

[in] Позиция элемента в массиве для изменения

element

[in] Новое значение элемента

Возвращаемое значение

true в случае успеха, false - если нет возможности изменить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Перемещает элемент из указанной позиции массива на указанное смещение.

```
bool  Shift(
   int  pos,        // позиция
   int  shift       // смещение
   )
```

Параметры

pos

[in] Позиция перемещаемого элемента массиве

shift

[in] Значение смещения (как положительное, так и отрицательное значение).

Возвращаемое значение

true в случае успеха, false - если нет возможности переместить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из указанной позиции массива.

```
bool  Delete(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Удаляет группу элементов из указанной позиции массива.

```
bool  DeleteRange(
   int  from,      // позиция первого элемента
   int  to         // позиция последнего элемента
   )
```

Параметры

from

[in] Позиция первого удаляемого элемента в массиве.

to

[in] Позиция последнего удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент из указанной позиции массива.

```
float  At(
   int  pos      // позиция
   ) const
```

Параметры

pos

[in] Позиция искомого элемента в массиве.

Возвращаемое значение

Значение элемента в случае успеха, FLT_MAX если была попытка получить элемент из не существующей позиции (при этом код последней ошибки ERR_OUT_OF_RANGE).

Примечание

Разумеется, FLT_MAX может быть и валидным значением элемента массива, поэтому, получив такое значение, всегда проверяйте код последней ошибки.

Пример:

```
//--- 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)
        {
         //--- ошибка чтения из массива
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- . . .
     }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const float&  src[]      // массив-источник
   ) const
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArrayconst

Копирует в массив элементы из другого массива.

```
bool  AssignArrayconst(
   const CArrayFloat*  src      // указатель на источник
   ) const
```

Параметры

src

[in] Указатель на экземпляр класса CArrayFloat-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Вставляет элемент в сортированный массив.

```
bool  InsertSort(
   float  element      // элемент для вставки
   )
```

Параметры

element

[in] Значение элемента для вставки в сортированный массив

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

Ищет элемент равный образцу в сортированном массиве.

```
int  Search(
   float  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Ищет элемент больше образца в сортированном массиве.

```
int  SearchGreat(
   float  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Ищет элемент меньше образца в сортированном массиве.

```
int  SearchLess(
   float  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Ищет элемент больше или равный образцу в сортированном массиве.

```
int  SearchGreatOrEqual(
   float  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Ищет элемент меньше или равный образцу в сортированном массиве.

```
int  SearchLessOrEqual(
   float  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Ищет первый элемент равный образцу в сортированном массиве.

```
int  SearchFirst(
   float  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Ищет последний элемент равный образцу в сортированном массиве.

```
int  SearchLast(
   float  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Ищет элемент равный образцу в массиве.

```
int  SearchLinear(
   float  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Примечание

Поиск производится последовательным перебором элементов массива (линейный поиск для несортированных массивов).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные массива в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       FileClose(file_handle);
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные массива из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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++)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      printf("Element[%d] = %f",i,array.At(i));
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа массива.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа массива (для CArrayFloat - 87).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayDouble

Класс CArrayDouble является классом динамического массива переменных типа double.

Описание

Класс CArrayDouble обеспечивает возможность работы с динамическим массивом переменных типа double. В классе реализованы возможности добавления/вставки/удаления элементов массива, сортировки массива, поисков в сортированном массиве. Кроме того, реализованы методы работы с файлом.

Декларация

```
class CArrayDouble : public CArray
```

Заголовок

```
#include <Arrays\ArrayDouble.mqh>
```

Иерархия наследования

CObject

CArray

CArrayDouble

Прямые потомки

CDoubleBuffer

Методы класса по группам

Атрибуты	
Delta	Устанавливает величину допуска сравнения
Управление памятью	
Reserve	Резервирует память для увеличения размеров массива
Resize	Устанавливает новый (меньший) размер массива
Shutdown	Очищает массив с полным освобождением памяти
Наполнение	
Add	Добавляет элемент в конец массива
AddArray	Добавляет в конец массива элементы из другого массива
AddArray	Добавляет в конец массива элементы из другого массива
Insert	Вставляет элемент в массив в указанную позицию

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
AssignArray	Копирует в массив элементы из другого массива
AssignArray	Копирует в массив элементы из другого массива
Изменение	
Update	Изменяет элемент в указанной позиции массива
Shift	Перемещает элемент из указанной позиции массива на указанное смещение
Удаление	
Delete	Удаляет элемент из указанной позиции массива
DeleteRange	Удаляет группу элементов из указанной позиции массива
Доступ	
At	Получает элемент из указанной позиции массива
Сравнение	
CompareArray	Сравнивает массив с другим массивом
CompareArray	Сравнивает массив с другим массивом
Поиск экстремумов	
Minimum	Получает индекс минимального значения в указанном диапазоне
Maximum	Получает индекс максимального значения в указанном диапазоне
Операции с сортированным массивом	
InsertSort	Вставляет элемент в сортированный массив
Search	Ищет элемент равный образцу в сортированном массиве
SearchGreat	Ищет элемент больше образца в сортированном массиве
SearchLess	Ищет элемент меньше образца в сортированном массиве
SearchGreatOrEqual	Ищет элемент больше или равный образцу в сортированном массиве

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
SearchLessOrEqual	Ищет элемент меньше или равный образцу в сортированном массиве
SearchFirst	Ищет первый элемент равный образцу в сортированном массиве
SearchLast	Ищет последний элемент равный образцу в сортированном массиве
SearchLinear	Ищет элемент равный образцу в массиве
Ввод/вывод	
virtual Save	Сохраняет данные массива в файле
virtual Load	Загружает данные массива из файла
virtual Type	Получает идентификатор типа массива

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Delta

Устанавливает допуск сравнения.

```
void  Delta(
   double  delta      // допуск
   )
```

Параметры

delta

[in] новое значение допуска сравнения.

Возвращаемое значение

Нет

Примечание

Допуск сравнения используется при поисках. Значения считаются равными, если их разница меньше или равна допуску. По умолчанию допуск равен 0.0.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

Резервирует память для увеличения размеров массива.

```
bool  Reserve(
   int  size      // количество
   )
```

Параметры

size

[in] Количество дополнительных элементов массива.

Возвращаемое значение

true в случае успеха, false - если была попытка запросить количество меньше или равное нулю, либо если массив увеличить не удалось.

Примечание

Для уменьшения фрагментации памяти, приращение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Устанавливает новый (меньший) размер массива.

```
bool  Resize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер массива.

Возвращаемое значение

true в случае успеха, false - если была попытка установить размер меньше нуля.

Примечание

Изменение размера массива позволяет оптимально использовать память. Лишние элементы справа теряются. Для уменьшения фрагментации памяти, изменение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Очищает массив с полным освобождением памяти.

```
bool  Shutdown()
```

Возвращаемое значение

true в случае успеха, false - если произошла ошибка.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в конец массива.

```
bool  Add(
   double  element      // элемент для добавления
   )
```

Параметры

element

[in] Значение элемента для добавления в массив.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const double&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const CArrayDouble*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayDouble-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в массив в указанную позицию.

```
bool  Insert(
   double  element,     // элемент для вставки
   int     pos          // позиция
   )
```

Параметры

element

[in] Значение элемента для вставки в массив

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанной позиции.

```
bool  InsertArray(
   const double&  src[],      // массив-источник
   int            pos         // позиция
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для вставки

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   CArrayDouble*  src,      // указатель на источник
   int            pos       // позиция
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayDouble-источник элементов для вставки.

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
      delete array;
      return;
     }
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const double&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const CArrayDouble*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayDouble-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- arrays is identical
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Изменяет элемент в указанной позиции массива.

```
bool  Update(
   int     pos,          // позиция
   double  element       // значение
   )
```

Параметры

pos

[in] Позиция элемента в массиве для изменения

element

[in] Новое значение элемента

Возвращаемое значение

true в случае успеха, false - если нет возможности изменить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Перемещает элемент из указанной позиции массива на указанное смещение.

```
bool  Shift(
   int  pos,        // позиция
   int  shift       // смещение
   )
```

Параметры

pos

[in] Позиция перемещаемого элемента массиве

shift

[in] Значение смещения (как положительное, так и отрицательное значение).

Возвращаемое значение

true в случае успеха, false - если нет возможности переместить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из указанной позиции массива.

```
bool  Delete(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Удаляет группу элементов из указанной позиции массива.

```
bool  DeleteRange(
   int  from,      // позиция первого элемента
   int  to         // позиция последнего элемента
   )
```

Параметры

from

[in] Позиция первого удаляемого элемента в массиве.

to

[in] Позиция последнего удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент из указанной позиции массива.

```
double  At(
   int  pos      // позиция
   ) const
```

Параметры

pos

[in] Позиция искомого элемента в массиве.

Возвращаемое значение

Значение элемента в случае успеха, DBL_MAX если была попытка получить элемент из не существующей позиции (при этом код последней ошибки ERR_OUT_OF_RANGE).

Примечание

Разумеется, DBL_MAX может быть и валидным значением элемента массива, поэтому, получив такое значение, всегда проверяйте код последней ошибки.

Пример:

```
//--- 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)
        {
         //--- ошибка чтения из массива
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- . . .
     }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const double&  src[]      // массив-источник
   ) const
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const CArrayDouble*  src      // указатель на источник
   ) const
```

Параметры

src

[in] Указатель на экземпляр класса CArrayDouble-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Minimum

Получает индекс минимального элемента массива в указанном интервале.

```
int  Minimum(
   int  start,      // стартовый индекс
   int  count       // количество
   ) const
```

Параметры

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

Возвращаемое значение

Индекс минимального элемента в указанном интервале.

© 2000-2025, MetaQuotes Ltd.

Maximum

Получает индекс максимального элемента массива в указанном интервале.

```
int  Maximum(
   int  start,      // стартовый индекс
   int  count       // количество
   ) const
```

Параметры

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

Возвращаемое значение

Индекс максимального элемента в указанном интервале.

© 2000-2025, MetaQuotes Ltd.

InsertSort

Вставляет элемент в сортированный массив.

```
bool  InsertSort(
   double  element      // элемент для вставки
   )
```

Параметры

element

[in] Значение элемента для вставки в сортированный массив

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

Ищет элемент равный образцу в сортированном массиве.

```
int  Search(
   double  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Ищет элемент больше образца в сортированном массиве.

```
int  SearchGreat(
   double  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Ищет элемент меньше образца в сортированном массиве.

```
int  SearchLess(
   double  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Ищет элемент больше или равный образцу в сортированном массиве.

```
int  SearchGreatOrEqual(
   double  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Ищет элемент меньше или равный образцу в сортированном массиве.

```
int  SearchLessOrEqual(
   double  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Ищет первый элемент равный образцу в сортированном массиве.

```
int  SearchFirst(
   double  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Ищет последний элемент равный образцу в сортированном массиве.

```
int  SearchLast(
   double  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Ищет элемент равный образцу в массиве.

```
int  SearchLinear(
   double  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Примечание

Поиск производится последовательным перебором элементов массива (линейный поиск для несортированных массивов).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные массива в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       FileClose(file_handle);
      }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные массива из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       printf("Element[%d] = %f",i,array.At(i));
      }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа массива.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа массива (для CArrayDouble - 87).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayString

Класс CArrayString является классом динамического массива переменных типа string.

Описание

Класс CArrayString обеспечивает возможность работы с динамическим массивом переменных типа string. В классе реализованы возможности добавления/вставки/удаления элементов массива, сортировки массива, поисков в сортированном массиве. Кроме того, реализованы методы работы с файлом.

Декларация

```
class CArrayString : public CArray
```

Заголовок

```
#include <Arrays\ArrayString.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayString

Методы класса по группам

Управление памятью	
Reserve	Резервирует память для увеличения размеров массива
Resize	Устанавливает новый (меньший) размер массива
Shutdown	Очищает массив с полным освобождением памяти
Наполнение	
Add	Добавляет элемент в конец массива
AddArray	Добавляет в конец массива элементы из другого массива
AddArray	Добавляет в конец массива элементы из другого массива
Insert	Вставляет элемент в массив в указанную позицию
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
AssignArray	Копирует в массив элементы из другого массива

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
AssignArray	Копирует в массив элементы из другого массива
Изменение	
Update	Изменяет элемент в указанной позиции массива
Shift	Перемещает элемент из указанной позиции массива на указанное смещение
Удаление	
Delete	Удаляет элемент из указанной позиции массива
DeleteRange	Удаляет группу элементов из указанной позиции массива
Доступ	
At	Получает элемент из указанной позиции массива
Сравнение	
CompareArray	Сравнивает массив с другим массивом
CompareArray	Сравнивает массив с другим массивом
Операции с сортированным массивом	
InsertSort	Вставляет элемент в сортированный массив
Search	Ищет элемент равный образцу в сортированном массиве
SearchGreat	Ищет элемент больше образца в сортированном массиве
SearchLess	Ищет элемент меньше образца в сортированном массиве
SearchGreatOrEqual	Ищет элемент больше или равный образцу в сортированном массиве
SearchLessOrEqual	Ищет элемент меньше или равный образцу в сортированном массиве
SearchFirst	Ищет первый элемент равный образцу в сортированном массиве
SearchLast	Ищет последний элемент равный образцу в сортированном массиве
SearchLinear	Ищет элемент равный образцу в массиве
Ввод/вывод	
virtual Save	Сохраняет данные массива в файле
virtual Load	Загружает данные массива из файла

© 2000-2025, MetaQuotes Ltd.

Управление памятью	
virtual Type	Получает идентификатор типа массива

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

Резервирует память для увеличения размеров массива.

```
bool  Reserve(
   int  size      // количество
   )
```

Параметры

size

[in] Количество дополнительных элементов массива.

Возвращаемое значение

true в случае успеха, false - если была попытка запросить количество меньше или равное нулю, либо если массив увеличить не удалось.

Примечание

Для уменьшения фрагментации памяти, приращение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Устанавливает новый (меньший) размер массива.

```
bool  Resize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер массива.

Возвращаемое значение

true в случае успеха, false - если была попытка установить размер меньше нуля.

Примечание

Изменение размера массива позволяет оптимально использовать память. Лишние элементы справа теряются. Для уменьшения фрагментации памяти, изменение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Очищает массив с полным освобождением памяти.

```
bool  Shutdown()
```

Возвращаемое значение

true в случае успеха, false - если произошла ошибка.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в конец массива.

```
bool  Add(
   string  element      // элемент для добавления
   )
```

Параметры

element

[in] Значение элемента для добавления в массив.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const string&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const CArrayString*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayString - источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в массив в указанную позицию.

```
bool  Insert(
   string  element,     // элемент для вставки
   int     pos          // позиция
   )
```

Параметры

element

[in] Значение элемента для вставки в массив

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   const string&  src[],      // массив-источник
   int             pos         // позиция
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для вставки

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   CArrayString*  src,      // указатель на источник
   int             pos      // позиция
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayString - источник элементов для вставки.

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
      delete array;
      return;
     }
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const string&  src[]      // массив-источник
   )
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const CArrayString*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayString - источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- arrays is identical
   //--- delete source array
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Изменяет элемент в указанной позиции массива.

```
bool  Update(
   int     pos,          // позиция
   string  element       // значение
   )
```

Параметры

pos

[in] Позиция элемента в массиве для изменения

element

[in] Новое значение элемента

Возвращаемое значение

true в случае успеха, false - если нет возможности изменить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Перемещает элемент из указанной позиции массива на указанное смещение.

```
bool  Shift(
   int  pos,        // позиция
   int  shift       // смещение
   )
```

Параметры

pos

[in] Позиция перемещаемого элемента массиве

shift

[in] Значение смещения (как положительное, так и отрицательное значение).

Возвращаемое значение

true в случае успеха, false - если нет возможности переместить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из указанной позиции массива.

```
bool  Delete(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Удаляет группу элементов из указанной позиции массива.

```
bool  DeleteRange(
   int  from,      // позиция первого элемента
   int  to         // позиция последнего элемента
   )
```

Параметры

from

[in] Позиция первого удаляемого элемента в массиве.

to

[in] Позиция последнего удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элементы.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент из указанной позиции массива.

```
string  At(
   int  pos      // позиция
   ) const
```

Параметры

pos

[in] Позиция искомого элемента в массиве.

Возвращаемое значение

Значение элемента в случае успеха, ""- если была попытка получить элемент из не существующей позиции (при этом код последней ошибки ERR_OUT_OF_RANGE).

Примечание

Разумеется, "" может быть и валидным значением элемента массива, поэтому, получив такое значение, всегда проверяйте код последней ошибки.

Пример:

```
//--- 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)
        {
         //--- ошибка чтения из массива
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- . . .
     }
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const string&  src[]      // массив-источник
   ) const
```

Параметры

src[]

[in] Ссылка на массив-источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArrays(
   const CArrayString*  src      // указатель на источник
   ) const
```

Параметры

src

[in] Указатель на экземпляр класса CArrayString - источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Вставляет элемент в сортированный массив.

```
bool  InsertSort(
   string  element      // элемент для вставки
   )
```

Параметры

element

[in] Значение элемента для вставки в сортированный массив

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

Ищет элемент равный образцу в сортированном массиве.

```
int  Search(
   string  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Ищет элемент больше образца в сортированном массиве.

```
int  SearchGreat(
   string  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Ищет элемент меньше образца в сортированном массиве.

```
int  SearchLess(
   string  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Ищет элемент больше или равный образцу в сортированном массиве.

```
int  SearchGreatOrEqual(
   string  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Ищет элемент меньше или равный образцу в сортированном массиве.

```
int  SearchLessOrEqual(
   string  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Ищет первый элемент равный образцу в сортированном массиве.

```
int  SearchFirst(
   string  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Ищет последний элемент равный образцу в сортированном массиве.

```
int  SearchLast(
   string  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

Ищет элемент равный образцу в массиве.

```
int  SearchLinear(
   string  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Примечание

Поиск производится последовательным перебором элементов массива (линейный поиск для несортированных массивов).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные массива в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       FileClose(file_handle);
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные массива из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       printf("Element[%d] = '%s'",i,array.At(i));
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа массива.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа массива (для CArrayString - 89).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayObj

Класс CArrayObj является классом динамического массива указателей на экземпляры класса CObject и его наследников.

Описание

Класс CArrayObj обеспечивает возможность работы с динамическим массивом указателей на экземпляры класса CObject и его наследников. Это дает возможность работы как с многомерными динамическими массивами примитивных типов данных, так и с более сложно организованными структурами данных.

В классе реализованы возможности добавления/вставки/удаления элементов массива, сортировки массива, поисков в сортированном массиве. Кроме того, реализованы методы работы с файлом.

Существуют определенные тонкости работы с классом CArrayObj.

Декларация

```
class CArrayObj : public CArray
```

Заголовок

```
#include <Arrays\ArrayObj.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj

Прямые потомки

CIndicators, CSeries

Методы класса по группам

Атрибуты	
FreeMode	Получает флаг управления памятью
FreeMode	Устанавливает флаг управления памятью
Управление памятью	
Reserve	Резервирует память для увеличения размеров массива
Resize	Устанавливает новый (меньший) размер массива
Shutdown	Очищает массив с полным освобождением памяти массива (но не элементов).
Создание нового элемента	

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
virtual CreateElement	Создает новый элемент массива в указанной позиции.
Наполнение	
Add	Добавляет элемент в конец массива
AddArray	Добавляет в конец массива элементы из другого массива
Insert	Вставляет элемент в массив в указанную позицию
InsertArray	Вставляет в массив элементы из другого массива с указанной позиции
AssignArray	Копирует в массив элементы из другого массива
Изменение	
Update	Изменяет элемент в указанной позиции массива
Shift	Перемещает элемент из указанной позиции массива на указанное смещение
Удаление	
Detach	Получает элемент из указанной позиции с удалением его из массива
Delete	Удаляет элемент из указанной позиции массива
DeleteRange	Удаляет группу элементов из указанной позиции массива
Clear	Удаляет все элементы массива без освобождения памяти массива
Доступ	
At	Получает элемент из указанной позиции массива
Сравнение	
CompareArray	Сравнивает массив с другим массивом
Операции с сортированным массивом	
InsertSort	Вставляет элемент в сортированный массив
Search	Ищет элемент равный образцу в сортированном массиве
SearchGreat	Ищет элемент больше образца в сортированном массиве
SearchLess	Ищет элемент меньше образца в сортированном массиве

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
SearchGreatOrEqual	Ищет элемент больше или равный образцу в сортированном массиве
SearchLessOrEqual	Ищет элемент меньше или равный образцу в сортированном массиве
SearchFirst	Ищет первый элемент равный образцу в сортированном массиве
SearchLast	Ищет последний элемент равный образцу в сортированном массиве
Ввод/вывод	
virtual Save	Сохраняет данные массива в файле
virtual Load	Загружает данные массива из файла
virtual Type	Получает идентификатор типа массива

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Практическое применение имеют массивы потомков класса CObject (в том числе все классы стандартной библиотеки).

Для примера рассмотрим вариант реализации двухмерного массива:

```
#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++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       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);
         }
       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;
  }
```

Тонкости работы

В классе реализован механизм управления динамической памятью, поэтому нужно быть очень внимательным при работе с элементами массива.

Механизм управления памятью можно включать/отключать при помощи метода FreeMode(bool). По умолчанию механизм включен.

Соответственно, есть два варианта работы с классом CArrayObj:

1. Механизм управления памятью включен. (по умолчанию)

В этом случае, CArrayObj берет на себя ответственность за освобождение памяти элементов после их удаления из массива. При этом программа пользователя не должна освобождать элементы массива.

Пример использования:

```
   int i;
//--- создаем массив
   CArrayObj *array=new CArrayObj;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- наполняем массив элементами
   for(i=0;i<10;i++) array.Add(new CObject);
//--- что-то делаем
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- действия с элементом
      . . .
     }
//--- удаляем массив вместе с элементами
   delete array;
```

2. Механизм управления памятью выключен.

В этом случае, CArrayObj не отвечает за освобождение памяти элементов после их удаления из массива. При этом программа пользователя должна освобождать элементы массива.

Пример использования:

```
   int i;
//--- создаем массив
   CArrayObj *array=new CArrayObj;
//--- выключаем механизм управления памятью
   array.FreeMode(false);
//--- наполняем массив элементами
   for(i=0;i<10;i++) array.Add(new CObject);
//--- что-то делаем
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- действия с элементом
      . . .
     }
//--- удаляем элементы массива
   while(array.Total()) delete array.Detach();
//--- удаляем пустой массив
   delete array;
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

Получает флаг управления памятью.

```
bool  FreeMode() const
```

Возвращаемое значение

Флаг управления памятью.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

Устанавливает флаг управления памятью.

```
void  FreeMode(
   bool  mode      // новый флаг
   )
```

Параметры

mode

[in] Новое значение флага управления памятью.

Возвращаемое значение

Нет.

Примечание

Установка флага управления памятью - важный момент в использовании класса CArrayObj. Так как элементами массива являются указатели на динамические объекты, важно определить, что делать с ними при удалении из массива.

Если флаг установлен, то при удалении элемента из массива, элемент автоматически удаляется оператором delete. Если же флаг не установлен, то подразумевается, что указатель на удаляемый объект остается еще где-то в программе пользователя и будет освобожден ею (программой) впоследствии.

Если программа пользователя сбрасывает флаг управления памятью, пользователь должен понимать свою ответственность за удаление элементов массива перед завершением программы, так как в противном случае остается неосвобожденной память, занятая элементами при их создании оператором new.

При больших объемах данных, это может привести, в конце концов, даже к нарушению работоспособности терминала. Если программа пользователя не сбрасывает флаг управления памятью, существует другой "подводный камень".

Использование указателей-элементов массива, сохраненных где-нибудь в локальных переменных, после удаления массива, приведет к критической ошибке и аварийному завершению программы пользователя. По умолчанию флаг управления памятью установлен, то есть класс массива сам отвечает за освобождение памяти элементов.

Пример:

```
//--- example for CArrayObj::FreeMode(bool)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      printf("Object create error");
      return;
     }
   //--- reset free mode flag
   array.FreeMode(false);
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

Резервирует память для увеличения размеров массива.

```
bool  Reserve(
   int  size      // количество
   )
```

Параметры

size

[in] Количество дополнительных элементов массива.

Возвращаемое значение

true в случае успеха, false - если была попытка запросить количество меньше или равное нулю, либо если массив увеличить не удалось.

Примечание

Для уменьшения фрагментации памяти, приращение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

Устанавливает новый (меньший) размер массива.

```
bool  Resize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер массива.

Возвращаемое значение

true в случае успеха, false - если была попытка установить размер меньше нуля.

Примечание

Изменение размера массива позволяет оптимально использовать память. Лишние элементы справа теряются. Память утерянных элементов освобождается или нет в зависимости от режима управления памятью.

Для уменьшения фрагментации памяти, изменение размеров массива производится с шагом ранее заданным через метод Step(int), либо 16 (по умолчанию).

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы массива без освобождения памяти массива.

```
void  Clear()
```

Возвращаемое значение

Нет.

Примечание

Если включен механизм управления памятью, память удаляемых элементов освобождается.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

Очищает массив с полным освобождением памяти массива (но не элементов).

```
bool  Shutdown()
```

Возвращаемое значение

true в случае успеха, false - если произошла ошибка.

Примечание

Если включен механизм управления памятью, память удаляемых элементов освобождается.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateElement

Создает новый элемент массива в указанной позиции.

```
bool  CreateElement(
   int  index      // позиция
   )
```

Параметры

index

[in] Позиция, в которой нужно создать новый элемент.

Возвращаемое значение

true в случае успеха, false - если нет возможности создать элемент.

Примечание

Метод CreateElement(int) в классе CArrayObj всегда возвращает false и не выполняет каких-либо действий. При необходимости в классе-наследнике, метод CreateElement(int) должен быть реализован.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в конец массива.

```
bool  Add(
   CObject*  element      // элемент для добавления
   )
```

Параметры

element

[in] Значение элемента для добавления в массив.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элемент.

Примечание

Элемент не добавится в массив, если в качестве параметра передать некорректный указатель (например NULL).

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

AddArray

Добавляет в конец массива элементы из другого массива.

```
bool  AddArray(
   const CArrayObj *  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayDouble - источник элементов для добавления.

Возвращаемое значение

true в случае успеха, false - если нет возможности добавить элементы.

Примечание

Добавление элементов из массива в массив фактически является копированием указателей. В связи с этим, при вызове метода, существует "подводный камень" - возможность существования указателя на динамический объект более чем в одной переменной.

```
//--- example
extern bool       make_error;
extern int        error;
extern CArrayObj *src;
//--- создаем новый экземпляр CArrayObj
//--- по умолчанию механизм управления памятью в нем включен
CArrayObj *array=new CArrayObj;
//--- добавляем (копируем) элементы из массива-источника
if(array!=NULL)
   bool result=array.AddArray(src);
if(make_error)
  {
   //--- совершаем ошибочные действия
   switch(error)
     {
      case 0:
         //--- удаляем массив-источник, не проверив его флаг управления памятью
         delete src;
         //--- результат:
         //--- возможно обращение
         //--- по "битому" указателю в массиве-приемнике
         break;
      case 1:
         //--- выключаем механизм управления памятью в массиве-источнике
         if(src.FreeMode()) src.FreeMode(false);
         //--- но не удаляем массив-источник
         //--- результат:
         //--- после удаления массива-приемника, возможно обращение
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- по "битому" указателю в массиве-источнике
         break;
      case 2:
         //--- выключаем механизм управления памятью в массиве-источнике
         src.FreeMode(false);
         //--- выключаем механизм управления памятью в массиве-приемнике
         array.FreeMode(false);
         //--- результат:
         //--- после завершения программы, получаем "утечку памяти"
         break;
     }
  }
else
  {
   //--- выключаем механизм управления памятью в массиве-источнике
   if(src.FreeMode()) src.FreeMode(false);
   //--- удаляем массив-источник
   delete src;
   //--- результат:
   //--- обращение к элементам массива-приемника будет корректным
   //--- удаление массива-приемника приведет к удалению его элементов
  }
```

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- reset free mode flag
   src.FreeMode(false);
   //--- fill source array
   //--- . . .
   //--- add another array
   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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в массив в указанную позицию.

```
bool  Insert(
   CObject*  element,      // элемент для вставки
   int       pos           // позиция
   )
```

Параметры

element

[in] Значение элемента для вставки в массив

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Примечание

Элемент не добавится в массив, если в качестве параметра передать некорректный указатель (например NULL).

Пример:

```
//--- 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;
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

Вставляет в массив элементы из другого массива с указанногй позиции.

```
bool  InsertArray(
   const CArrayObj*  src,      // указатель на источник
   int               pos       // позиция
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayObj-источник элементов для вставки.

pos

[in] Позиция в массиве для вставки

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элементы.

Примечание

См. CArrayObj::AddArray(const CArrayObj*).

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- . . .
   //--- insert another array
   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;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

Копирует в массив элементы из другого массива.

```
bool  AssignArray(
   const CArrayObj*  src      // указатель на источник
   )
```

Параметры

src

[in] Указатель на экземпляр класса CArrayObj - источник элементов для копирования.

Возвращаемое значение

true в случае успеха, false - если нет возможности скопировать элементы.

Примечание

Если перед вызовом AssignArray массив приемник не пуст, все его элементы будут удалены из массива и, если установлен флаг управления памятью, память удаленных элементов будет освобождена. Массив-приемник становится точной копией массива источника. Дополнительно см. CArrayObj::AddArray(const CArrayObj*).

Пример:

```
//--- 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
   //--- . . .
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- assign another array
   if(!array.AssignArray(src))
     {
      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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Изменяет элемент в указанной позиции массива.

```
bool  Update(
   int       pos,          // позиция
   CObject*  element       // значение
   )
```

Параметры

pos

[in] Позиция элемента в массиве для изменения

element

[in] Новое значение элемента

Возвращаемое значение

true в случае успеха, false - если нет возможности изменить элемент.

Примечание

Элемент не изменится, если в качестве параметра передать некорректный указатель (например NULL). Если включен механизм управления памятью, память замещаемого элемента освобождается.

Пример:

```
//--- 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;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

Перемещает элемент из указанной позиции массива на указанное смещение.

```
bool  Shift(
   int  pos,        // позиция
   int  shift       // смещение
   )
```

Параметры

pos

[in] Позиция перемещаемого элемента массиве

shift

[in] Значение смещения (как положительное, так и отрицательное значение).

Возвращаемое значение

true в случае успеха, false - если нет возможности переместить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Detach

Изымает элемент из указанной позиции массива.

```
CObject*  Detach(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция изымаемого элемента в массиве.

Возвращаемое значение

Указатель на изъятый элемент в случае успеха, NULL - если нет возможности изъять элемент.

Примечание

При изъятии из массива элемент не удалится при любом состоянии флага управления памятью. Указатель на изъятый из массива элемент небходимо освободить после использования.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из указанной позиции массива.

```
bool  Delete(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Примечание

Если включен механизм управления памятью, память удаляемых элементов освобождается.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

Удаляет группу элементов из указанной позиции массива.

```
bool  DeleteRange(
   int  from,      // позиция первого элемента
   int  to         // позиция последнего элемента
   )
```

Параметры

from

[in] Позиция первого удаляемого элемента в массиве.

to

[in] Позиция последнего удаляемого элемента в массиве.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элементы.

Примечание

Если включен механизм управления памятью, память удаляемых элементов освобождается.

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент из указанной позиции массива.

```
CObject*  At(
   int  pos      // позиция
   )
```

Параметры

pos

[in] Позиция искомого элемента в массиве.

Возвращаемое значение

Значение элемента в случае успеха, NULL- если была попытка получить элемент из не существующей позиции.

Пример:

```
//--- 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)
        {
         //--- ошибка чтения из массива
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
      //--- . . .
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

CompareArray

Сравнивает массив с другим массивом.

```
bool  CompareArray(
   const CArrayObj*  src      // указатель на источник
   ) const
```

Параметры

src

[in] Указатель на экземпляр класса CArrayObj - источник элементов для сравнения.

Возвращаемое значение

true в случае массивы равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

Вставляет элемент в сортированный массив.

```
bool  InsertSort(
   CObject*  element      // элемент для вставки
   )
```

Параметры

element

[in] Значение элемента для вставки в сортированный массив

Возвращаемое значение

true в случае успеха, false - если нет возможности вставить элемент.

Примечание

Элемент не добавится в массив, если в качестве параметра передать некорректный указатель (например NULL).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

Ищет элемент равный образцу в сортированном массиве.

```
int  Search(
   CObject*  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

Ищет элемент больше образца в сортированном массиве.

```
int  SearchGreat(
   CObject*  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

Ищет элемент меньше образца в сортированном массиве.

```
int  SearchLess(
   CObject*  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

Ищет элемент больше или равный образцу в сортированном массиве.

```
int  SearchGreatOrEqual(
   CObject*  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

Ищет элемент меньше или равный образцу в сортированном массиве.

```
int  SearchLessOrEqual(
   CObject*  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

Ищет первый элемент равный образцу в сортированном массиве.

```
int  SearchFirst(
   CObject*  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

Ищет последний элемент равный образцу в сортированном массиве.

```
int  SearchLast(
   CObject*  element      // образец
   ) const
```

Параметры

element

[in] Образец элемента для поиска в массиве.

Возвращаемое значение

Позиция найденного элемента в случае успеха, -1 - если элемент не найден.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные массива в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого при помощи функции FileOpen(...) бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные массива из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Примечание

При чтении из файла элементов массива, для создания каждого элемента вызывается метод CArrayObj::CreateElement(int).

Пример:

```
//--- 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;
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      FileClose(file_handle);
     }
   //--- use arrays elements
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа массива.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа массива (для CArrayObj - 7778).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CList

Класс CList является классом динамического списка экземпляров класса CObject и его наследников.

Описание

Класс CList обеспечивает возможность работы со списком экземпляров класса CObject и его наследников. В классе реализованы возможности добавления/вставки/удаления элементов списка, сортировки списка, поисков в сортированном списке. Кроме того, реализованы методы работы с файлом.

Существуют определенные тонкости работы с классом CList. В классе реализован механизм управления динамической памятью, поэтому нужно быть очень внимательным при работе с элементами списка.

Тонкости работы с механизмом управления памятью аналогичны описанным в CArrayObj.

Декларация

```
class CList : public CObject
```

Заголовок

```
#include <Arrays\List.mqh>
```

Иерархия наследования

CObject
  CList

Методы класса по группам

Атрибуты	
FreeMode	Получает флаг управления памятью при удалении элементов списка.
FreeMode	Устанавливает флаг управления памятью при удалении элементов списка
Total	Получает количество элементов в списке
IsSorted	Получает флаг сортированности списка
SortMode	Получает вариант сортировки
Создание нового элемента	
virtual CreateElement	Создает новый элемент списка
Наполнение	
Add	Добавляет элемент в конец списка

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
Insert	Вставляет элемент в список в указанную позицию
Удаление	
DetachCurrent	Изымает элемент из текущей позиции списка, не удаляя его “физически"
DeleteCurrent	Удаляет элемент из текущей позиции списка
Delete	Удаляет элемент из указанной позиции списка
Clear	Удаляет все элементы списка
Навигация	
IndexOf	Получает индекс указанного элемента списка
GetNodeAtIndex	Получает элемент из указанной позиции списка
GetFirstNode	Получает первый элемент списка
GetPrevNode	Получает предыдущий элемент списка
GetCurrentNode	Получает текущий элемент списка
GetNextNode	Получает следующий элемент списка
GetLastNode	Получает последний элемент списка
Перемещение элементов	
Sort	Сортирует список
MoveToIndex	Перемещает текущий элемент списка в указанную позицию
Exchange	Меняет элементы списка местами
Сравнение	
CompareList	Сравнивает список с другим списком
Поиск	
Search	Ищет элемент, равный образцу, в сортированном списке
Ввод/вывод	
virtual Save	Сохраняет данные списка в файл
virtual Load	Загружает данные списка из файла
virtual Type	Получает идентификатор типа списка

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

© 2000-2025, MetaQuotes Ltd.

FreeMode

Получает флаг управления памятью при удалении элементов списка.

```
bool  FreeMode() const
```

Возвращаемое значение

Флаг управления памятью.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

Устанавливает флаг управления памятью при удалении элементов списка.

```
void  FreeMode(
   bool  mode      // новое значение
   )
```

Параметры

mode

[in] Новое значение флага управления памятью.

Примечание

Установка флага управления памятью - важный момент в использовании класса CList. Так как элементами списка являются указатели на динамические объекты, важно определить, что делать с ними при удалении из списка.

Если флаг установлен, то при удалении элемента из списка, элемент автоматически удаляется оператором delete. Если же флаг не установлен, то подразумевается, что указатель на удаляемый объект остается еще где-то в программе пользователя и будет освобожден ею (программой) впоследствии.

Если программа пользователя сбрасывает флаг управления памятью, пользователь должен понимать свою ответственность за удаление элементов списка перед завершением программы, так как в противном случае остается неосвобожденной память, занятая элементами при их создании оператором new.

При больших объемах данных, это может привести, в конце концов, даже к нарушению работоспособности терминала.

Если программа пользователя не сбрасывает флаг управления памятью, существует другой "подводный камень". Использование указателей-элементов списка, сохраненных где-нибудь в локальных переменных, после удаления списка, приведет к критической ошибке и аварийному завершению программы пользователя. По умолчанию флаг управления памятью установлен, то есть класс списка сам отвечает за освобождение памяти элементов.

Пример:

```
//--- example for CList::FreeMode(bool)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- reset free mode flag
   list.FreeMode(false);
   //--- use list
   //--- . . .
   //--- delete list
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Total

Получает количество элементов в списке.

```
int  Total() const
```

Возвращаемое значение

Количество элементов в списке.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

IsSorted

Получает флаг сортированности списка.

```
bool  IsSorted(
   int  mode=0       // вариант сортировки
   ) const
```

Параметры

mode=0

[in] Проверяемый вариант сортировки

Возвращаемое значение

Флаг сортированности списка. Если список сортированный по указанному варианту - true, иначе false.

Примечание

Флаг сортированности списка нельзя изменить непосредственно. Флаг устанавливается методом Sort(int) и сбрасывается любыми методами добавления/вставки.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

SortMode

Получает вариант сортировки.

```
int  SortMode() const
```

Возвращаемое значение

Вариант сортировки, либо -1 если список не сортирован.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateElement

Создает новый элемент списка.

```
CObject*  CreateElement()
```

Возвращаемое значение

Указатель на вновь созданный элемент в случае успеха, NULL - если нет возможности создать элемент.

Примечание

Метод CreateElement() в классе CList всегда возвращает NULL и не выполняет каких-либо действий. При необходимости в классе-наследнике, метод CreateElement() должен быть реализован.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в конец списка.

```
int  Add(
   CObject*  element      // элемент для добавления
   )
```

Параметры

element

[in] значение элемента для добавления в список.

Возвращаемое значение

Индекс добавленного элемента, либо -1 в случае ошибки.

Примечание

Элемент не добавится в список, если в качестве параметра передать некорректный указатель (например NULL).

Пример:

```
//--- 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;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в список в указанную позицию.

```
int  Insert(
   CObject*  element,      // элемент для вставки
   int       pos           // позиция
   )
```

Параметры

element

[in] значение элемента для вставки в список

pos

[in] позиция в списке для вставки

Возвращаемое значение

Индекс вставленного элемента, либо -1 в случае ошибки.

Примечание

Элемент не добавится в список, если в качестве параметра передать некорректный указатель (например NULL).

Пример:

```
//--- 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;
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- use list
   //--- . . .
   //--- delete list
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

DetachCurrent

Изымает элемент из текущей позиции списка не удаляя его "физически".

```
CObject*  DetachCurrent()
```

Возвращаемое значение

Указатель на изъятый элемент в случае успеха, NULL - если нет возможности изъять элемент.

Примечание

При изъятии из списка, элемент не удалится при любом состоянии флага управления памятью. Указатель на изъятый из списка элемент необходимо освободить после использования.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteCurrent

Удаляет элемент из текущей позиции списка.

```
bool  DeleteCurrent()
```

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Примечание

Если включен механизм управления памятью, память удаляемого элемента освобождается.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из указанной позиции списка.

```
bool  Delete(
   int  pos      // позиция
   )
```

Параметры

pos

[in] позиция удаляемого элемента в списке.

Возвращаемое значение

true в случае успеха, false - если нет возможности удалить элемент.

Примечание

Если включен механизм управления памятью, память удаляемого элемента освобождается.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы списка.

```
void  Clear()
```

Примечание

Если включен механизм управления памятью, память удаляемых элементов освобождается.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

IndexOf

Получает индекс указанного элемента списка.

```
int  IndexOf(
   CObject*  element      // указатель на элемент
   )
```

Параметры

element

[in] указатель на элемент списка.

Возвращаемое значение

Индекс элемента списка, либо -1.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetNodeAtIndex

Получает элемент из указанной позиции списка.

```
CObject*  GetNodeAtIndex(
   int  pos      // позиция
   )
```

Параметры

pos

[in] позиция элемента в списке.

Возвращаемое значение

указатель на элемент в случае успеха, NULL - если нет возможности получить указатель.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetFirstNode

Получает первый элемент списка.

```
CObject*  GetFirstNode()
```

Возвращаемое значение

Указатель на первый элемент в случае успеха, NULL - если нет возможности получить указатель.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetPrevNode

Получает предыдущий элемент списка.

```
CObject*  GetPrevNode()
```

Возвращаемое значение

Указатель на предыдущий элемент в случае успеха, NULL - если нет возможности получить указатель.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetCurrentNode

Получает текущий элемент списка.

```
CObject*  GetCurrentNode()
```

Возвращаемое значение

Указатель на текущий элемент в случае успеха, NULL - если нет возможности получить указатель.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetNextNode

Получает следующий элемент списка.

```
CObject*  GetNextNode()
```

Возвращаемое значение

Указатель на следующий элемент в случае успеха, NULL - если нет возможности получить указатель.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetLastNode

Получает последний элемент списка.

```
CObject*  GetLastNode()
```

Возвращаемое значение

Указатель на последний элемент в случае успеха, NULL - если нет возможности получить указатель.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Sort

Сортирует список.

```
void  Sort(
   int  mode      // вариант сортировки
   )
```

Параметры

mode

[in] Вариант сортировки.

Возвращаемое значение

Нет.

Примечание

Сортировка списка производится всегда по возрастанию.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

MoveToIndex

Перемещает текущий элемент списка в указанную позицию.

```
bool  MoveToIndex(
   int  pos      // позиция
   )
```

Параметры

pos

[in] позиция в списке для перемещения.

Возвращаемое значение

true в случае успеха, false - если нет возможности переместить элемент.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Exchange

Меняет элементы списка местами.

```
bool  Exchange(
   CObject*  node1,      // элемент списка
   CObject*  node2       // элемент списка
   )
```

Параметры

node1

[in] элемент списка

node2

[in] элемент списка

Возвращаемое значение

true в случае успеха, false - если нет возможности поменять элементы местами.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareList

Сравнивает список с другим списком.

```
bool  CompareList(
   CList*  list      // с кем сравниваем
   )
```

Параметры

list

[in] указатель на экземпляр класса CList-источник элементов для сравнения.

Возвращаемое значение

true в случае если списки равны, false - если нет.

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

Ищет элемент равный образцу, в сортированном списке.

```
CObject*  Search(
   CObject*  element      // образец
   )
```

Параметры

element

[in] образец элемента для поиска в списке.

Возвращаемое значение

Указатель на найденный элемента в случае успеха, NULL - если элемент не найден.

Пример:

```
//--- 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
```

© 2000-2025, MetaQuotes Ltd.

```
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные списка в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл бинарного файла, открытого ранее при помощи функции FileOpen(...).

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- delete list
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает данные списка из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл бинарного файла, открытого ранее при помощи функции FileOpen(...).

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Примечание

При чтении из файла элементов списка, для создания каждого элемента вызывается метод CList::CreateElement().

Пример:

```
//--- 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;
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      FileClose(file_handle);
     }
   //--- use list elements
   //--- . . .
   //--- delete list
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа списка.

```
virtual int  Type()
```

Возвращаемое значение

Идентификатор типа списка (для CList - 7779).

Пример:

```
//--- 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;
  }
```

© 2000-2025, MetaQuotes Ltd.

CTreeNode

Класс CTreeNode является классом узла двоичного дерева CTree.

Описание

Класс CTreeNode обеспечивает возможность работы с узлами двоичного дерева CTree. В классе реализованы возможности навигации по дереву. Кроме того, реализованы методы работы с файлом.

Декларация

```
class CArrayObj : public CObject
```

Заголовок

```
#include <Arrays\TreeNode.mqh>
```

Иерархия наследования

CObject

CTreeNode

Прямые потомки

CTree

Методы класса по группам

Атрибуты	
Owner	Получает/устанавливает указатель узла-владельца
Left	Получает/устанавливает указатель левого узла
Right	Получает/устанавливает указатель правого узла
Balance	Получает баланс узла
BalanceL	Получает баланс левой подветви узла
BalanceR	Получает баланс правой подветви узла
Создание нового элемента	
CreateSample	Создает новый экземпляр узла
Сравнение	
RefreshBalance	Пересчитывает баланс узла
Поиск	
GetNext	Получает указатель следующего узла
Ввод/вывод	

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
SaveNode	Сохраняет данные узла в файл
LoadNode	Загружает данные узла из файла
virtual Type	Получает идентификатор типа узла

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Compare

Практическое применение имеют деревья потомков класса CTreeNode.

Потомок класса CTreeNode должен иметь переопределенные методы: CreateSample, создающий новый экземпляр класса-потомка CTreeNode, Compare, сравнивающий значения ключевых полей класса-потомка CTreeNode, Type (при необходимости идентификации узла), SaveNode и LoadNode (при необходимости работы с файлом).

Рассмотрим пример класса-потомка 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>
//+------------------------------------------------------------------+
//| Описываем класс CMyTreeNode производный от CTreeNode.            |
//+------------------------------------------------------------------+
//| Класс CMyTreeNode.                                               |
//| Назначение: Класс элемента двоичного дерева.                     |
//|             Потомок класса CTreeNode.                            |
//+------------------------------------------------------------------+
class CMyTreeNode : public CTreeNode
  {
protected:
   //--- данные пользователя
   long              m_long;           // ключевое поле типа long
   double            m_double;         // переменная пользователя типа double
   string            m_string;         // переменная пользователя типа string
   datetime          m_datetime;       // переменная пользователя типа datetime

public:
                     CMyTreeNode();
   //--- методы доступа к данным пользователя
   long              GetLong(void)                { return(m_long); }
   void              SetLong(long value)          { m_long=value;   }
```

© 2000-2025, MetaQuotes Ltd.

```
   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;  }
   //--- методы работы с файлами
   virtual bool      Save(int file_handle);
   virtual bool      Load(int file_handle);
protected:
   virtual int       Compare(const CObject *node,int mode);
   //--- метод создания экземпляра класса
   virtual CTreeNode* CreateSample();
  };
//+------------------------------------------------------------------+
//| Конструктор класса CMyTreeNode.                                  |
//| INPUT:  нет.                                                     |
//| OUTPUT: нет.                                                     |
//| REMARK: нет.                                                     |
//+------------------------------------------------------------------+
void CMyTreeNode::CMyTreeNode()
  {
//--- инициализация данных пользователя
   m_long        =0;
   m_double      =0.0;
   m_string      ="";
   m_datetime    =0;
  }
//+------------------------------------------------------------------+
//| Сравнение с другим узлом дерева по указанному алгоритму.         |
//| INPUT:  node - элемент массива для сравнения,                    |
//|         mode - идентификатор алгоритма сравнения.                |
//| OUTPUT: результат сравнения (>0,0,<0).                           |
//| REMARK: нет.                                                     |
//+------------------------------------------------------------------+
int CMyTreeNode::Compare(const CObject *node,int mode)
  {
//--- параметр mode игнорируется, т.к. агоритм построения дерева единственный
   int res=0;
//--- явное преобразование типа
   CMyTreeNode *n=node;
   res=(int)(m_long-n.m_long);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Создание нового экземпляра класса.                               |
//| INPUT:  нет.                                                     |
//| OUTPUT: указатель на новый экземпляр класса CMyTreeNode.         |
```

© 2000-2025, MetaQuotes Ltd.

```
//| REMARK: нет.                                                     |
//+------------------------------------------------------------------+
CTreeNode* CMyTreeNode::CreateSample()
  {
   CMyTreeNode *result=new CMyTreeNode;
//---
   return(result);
  }
//+------------------------------------------------------------------+
//| Запись данных узла дерева в файл.                                |
//| INPUT:  file_handle -хендл ранее открытого для записи файла.     |
//| OUTPUT: true если OK, иначе false.                               |
//| REMARK: нет.                                                     |
//+------------------------------------------------------------------+
bool CMyTreeNode::Save(int file_handle)
  {
   uint i=0,len;
//--- проверки
   if(file_handle<0) return(false);
//--- собственно запись данных пользователя
//--- запись переменной пользователя типа long
   if(FileWriteLong(file_handle,m_long)!=sizeof(long))          return(false);
//--- запись переменной пользователя типа double
   if(FileWriteDouble(file_handle,m_double)!=sizeof(double))    return(false);
//--- запись переменной пользователя типа string
   len=StringLen(m_string);
//--- запись длины строки
   if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE)   return(false);
//--- запись собственно строки
   if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- запись переменной пользователя типа datetime
   if(FileWriteLong(file_handle,m_datetime)!=sizeof(long))      return(false);
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Чтение данных узла дерева из файла.                              |
//| INPUT:  file_handle -хендл ранее открытого для чтения файла.     |
//| OUTPUT: true если OK, иначе false.                               |
//| REMARK: нет.                                                     |
//+------------------------------------------------------------------+
bool CMyTreeNode::Load(int file_handle)
  {
   uint i=0,len;
//--- проверки
   if(file_handle<0) return(false);
//--- собственно чтение
   if(FileIsEnding(file_handle)) return(false);
//--- чтение переменной пользователя типа char
```

© 2000-2025, MetaQuotes Ltd.

```
//--- чтение переменной пользователя типа long
   m_long=FileReadLong(file_handle);
//--- чтение переменной пользователя типа double
   m_double=FileReadDouble(file_handle);
//--- чтение переменной пользователя типа string
//--- чтение длины строки
   len=FileReadInteger(file_handle,INT_VALUE);
//--- чтение собственно строки
   if(len!=0) m_string=FileReadString(file_handle,len);
   else       m_string="";
//--- чтение переменной пользователя типа datetime
   m_datetime=FileReadLong(file_handle);
//---
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

Owner

Получает указатель узла-владельца.

```
CTreeNode*  Owner()
```

Возвращаемое значение

Указатель узла-владельца.

Owner

Устанавливает указатель узла-владельца.

```
void  Owner(
   CTreeNode*  node      // узел
   )
```

Параметры

node

[in] Новое значение указателя узла-владельца.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Left

Получает указатель левого узла.

```
CTreeNode*  Left()
```

Возвращаемое значение

Указатель левого узла.

Left

Устанавливает указатель левого узла.

```
void  Left(
   CTreeNode*  node      // узел
   )
```

Параметры

node

[in] Новое значение указателя левого узла.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Right

Получает указатель правого узла.

```
CTreeNode*  Right()
```

Возвращаемое значение

Указатель правого узла.

Right

Устанавливает указатель правого узла.

```
void  Right(
   CTreeNode*  node      // узел
   )
```

Параметры

node

[in] Новое значение указателя правого узла.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Balance

Получает баланс узла.

```
int  Balance() const
```

Возвращаемое значение

Баланс узла.

© 2000-2025, MetaQuotes Ltd.

BalanceL

Получает баланс левой подветви узла.

```
int  BalanceL() const
```

Возвращаемое значение

Баланс левой подветви узла.

© 2000-2025, MetaQuotes Ltd.

BalanceR

Получает баланс правой подветви узла.

```
int  BalanceR() const
```

Возвращаемое значение

Баланс правой подветви узла.

© 2000-2025, MetaQuotes Ltd.

CreateSample

Создает новый экземпляр узла.

```
virtual CTreeNode*  CreateSample()
```

Возвращаемое значение

Указатель нового экземпляра узла, либо NULL.

© 2000-2025, MetaQuotes Ltd.

RefreshBalance

Пересчитывает баланс узла.

```
int  RefreshBalance()
```

Возвращаемое значение

Баланс узла.

© 2000-2025, MetaQuotes Ltd.

GetNext

Получает указатель следующего узла.

```
CTreeNode*  GetNext(
   CTreeNode*  node      // узел
   )
```

Параметры

node

[in] Узел начала поиска.

Возвращаемое значение

Указатель следующего узла.

© 2000-2025, MetaQuotes Ltd.

SaveNode

Записывает данные узла в файл.

```
bool  SaveNode(
   int  file_handle      // хэндл
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого для записи бинарного файла.

Возвращаемое значение

true в случае удачи, иначе false.

© 2000-2025, MetaQuotes Ltd.

LoadNode

Читает данные узла из файла.

```
bool  LoadNode(
   int          file_handle,     // хэндл
   CTreeNode*   main             // узел
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого для чтения бинарного файла.

main

[in] Узел для данных.

Возвращаемое значение

true в случае удачи, иначе false.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа узла.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа узла.

© 2000-2025, MetaQuotes Ltd.

CTree

Класс CTree является классом двоичного дерева экземпляров класса CTreeNode и его наследников.

Описание

Класс CTree обеспечивает возможность работы с двоичным деревом экземпляров класса CTreeNode и его наследников. В классе реализованы возможности добавления/вставки/удаления элементов дерева, поиска в дереве. Кроме того, реализованы методы работы с файлом.

Следует отметить что, в классе CTree не реализован механизм управления динамической памятью (в отличии от классов CList и CArrayObj). Все узлы дерева удаляются с освобождением памяти.

Декларация

```
class CTree : public CTreeNode
```

Заголовок

```
#include <Arrays\Tree.mqh>
```

Иерархия наследования

CObject
  CTreeNode
    CTree

Методы класса по группам

Атрибуты	
Root	Получает корневой узел дерева
Создание нового элемента	
virtual CreateElement	Создает новый экземпляр узла
Наполнение	
Insert	Добавляет узел в дерево
Удаление	
Detach	Изымает указанный узел из дерева
Delete	Удаляет указанный узел из дерева
Clear	Удаляет все узлы дерева
Поиск	
Find	Ищет в дереве узел по образцу
Ввод/вывод	

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
virtual Save	Сохраняет данные дерева в файл
virtual Load	Загружает данные дерева из файла
virtual Type	Получает идентификатор типа дерева

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CTreeNode

Parent, Parent, Left, Left, Right, Right, Balance, BalanceL, BalanceR, RefreshBalance, GetNext, SaveNode, LoadNode

Практическое применение имеют деревья потомков класса CTreeNode - потомки класса CTree.

Потомок класса CTree должен иметь переопределенный метод CreateElement, создающий новый экземпляр класса-потомка CTreeNode.

Рассмотрим пример класса-потомка CTree.

```
//+------------------------------------------------------------------+
//|                                                      MyTree.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                              https://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;
//+------------------------------------------------------------------+
//| Описываем класс CMyTree производный от CTree.                    |
//+------------------------------------------------------------------+
//| Класс CMyTree.                                                   |
//| Назначение: Построение и навигация двоичного дерева поиска.      |
//+------------------------------------------------------------------+
class CMyTree : public CTree
  {
public:
   //--- методы поиска в дереве по данным пользователя
   CMyTreeNode*         FindByLong(long find_long);
   //--- метод создания элемента дерева
   virtual CTreeNode *CreateElement();
  };
//---
CMyTree MyTree;
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Создание нового узла дерева.                                     |
//| INPUT:  нет.                                                     |
//| OUTPUT: указатель на новый узел дерева если OK, либо NULL.       |
//| REMARK: нет.                                                     |
//+------------------------------------------------------------------+
CTreeNode *CMyTree::CreateElement()
  {
   CMyTreeNode *node=new CMyTreeNode;
//---
   return(node);
  }
//+------------------------------------------------------------------+
//| Поиск элемента в списке по значению m_long.                      |
//| INPUT:  find_long - искомое значение.                            |
//| OUTPUT: указатель найденного элемента списка, либо NULL.         |
//| REMARK: нет.                                                     |
//+------------------------------------------------------------------+
CMyTreeNode* CMyTree::FindByLong(long find_long)
  {
   CMyTreeNode *res=NULL;
   CMyTreeNode *node;
//--- создаем узел дерева для передачи параметра поиска
   node=new CMyTreeNode;
   if(node==NULL) return(NULL);
   node.SetLong(find_long);
//---
   res=Find(node);
   delete node;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| скрипт "тестирование класса CMyTree"                             |
//+------------------------------------------------------------------+
//---  массив для инициализации строк
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; //--- временный указатель на экземпляр класса CMyTreeNode
//---
   printf("Start test %s.",__FILE__);
//--- Наполняем MyTree экземплярами класса MyTreeNode в количестве extCountedNodes.
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       node=MyTree.CreateElement();
       if(node==NULL)
         {
          //--- аварийный выход
          printf("%s (%4d): create error",__FILE__,__LINE__);
          return(__LINE__);
         }
       NodeSetData(node,i);
       node.SetLong(i);
       MyTree.Insert(node);
      }
   end_time=GetTickCount();
   printf("Время заполнения MyTree %d мс.",end_time-beg_time);
//--- Создаем временное дерево TmpMyTree.
   CMyTree TmpMyTree;
//--- Изымаем 50% элементов из дерева (все четные)
//--- и добавляем их во временное дерево 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("Время удаления %d элементов из MyTree %d мс.",extCountedNodes/2,end_time-be
//--- Возвращаем изятое обратно
   node=TmpMyTree.Root();
   while(node!=NULL)
     {
      if(TmpMyTree.Detach(node)) MyTree.Insert(node);
      node=TmpMyTree.Root();
     }
//--- Проверяем работу метода 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))
        {
         //--- ошибка записи в файл
         //--- аварийный выход
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- уходя, закройте файл !!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- Проверяем работу метода Load(int file_handle);
   file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!TmpMyTree.Load(file_handle))
        {
         //--- ошибка чтения из файла
         //--- аварийный выход
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- уходя, закройте файл !!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//---
   MyTree.Clear();
   TmpMyTree.Clear();
//---
   printf("End test %s. OK!",__FILE__);
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| Функция вывода содержимого node в журнал                         |
//+------------------------------------------------------------------+
void NodeToLog(CMyTreeNode *node)
  {
   printf("   %I64d,%f,'%s','%s'",
              node.GetLong(),node.GetDouble(),
              node.GetString(),TimeToString(node.GetDateTime()));
  }
//+------------------------------------------------------------------+
//| Функция "наполнения" node случайными значениями                  |
//+------------------------------------------------------------------+
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());
     }
   node.SetString(str_array[mode%10]);
```

© 2000-2025, MetaQuotes Ltd.

```
   node.SetDateTime(10000*mode);
  }
```

© 2000-2025, MetaQuotes Ltd.

Root

Получает корневой узел дерева.

```
CTreeNode*  Root() const
```

Возвращаемое значение

Указатель корневого узла дерева.

© 2000-2025, MetaQuotes Ltd.

CreateElement

Создает новый экземпляр узла.

```
virtual CTreeNode*  CreateElement()
```

Возвращаемое значение

Указатель нового экземпляра узла, либо NULL.

© 2000-2025, MetaQuotes Ltd.

Insert

Добавляет узел в дерево.

```
CTreeNode*  Insert(
   CTreeNode*  new_node      // узел
   )
```

Параметры

new_node

[in] Указатель узла для вставки в дерево.

Возвращаемое значение

Указатель узла-владельца, либо NULL.

© 2000-2025, MetaQuotes Ltd.

Detach

Изымает указанный узел из дерева.

```
bool  Detach(
   CTreeNode*  node      // узел
   )
```

Параметры

node

[in] Указатель узла для изъятия.

Возвращаемое значение

true-в случае удачи, иначе false.

Примечание

После изъятия из дерева указатель узла не освобождается. Дерево балансируется.

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет указанный узел из дерева.

```
bool  Delete(
   CTreeNode*  node      // узел
   )
```

Параметры

node

[in] Указатель узла для удаления.

Возвращаемое значение

true-в случае удачи, иначе false.

Примечание

После удаления из дерева указатель узла освобождается. Дерево балансируется.

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все узлы дерева.

```
void  Clear()
```

Возвращаемое значение

Нет.

Примечание

После удаления из дерева указатели узлов освобождаются.

© 2000-2025, MetaQuotes Ltd.

Find

Ищет в дереве узел по образцу.

```
CTreeNode*  Find(
   CTreeNode*  node      // узел
   )
```

Параметры

node

[in] Узел, содержащий данные-образец поиска.

Возвращаемое значение

Указатель найденного узла, либо NULL.

© 2000-2025, MetaQuotes Ltd.

Save

Записывает данные дерева в файл.

```
virtual bool  Save(
   int  file_handle      // хэндл
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого для записи бинарного файла.

Возвращаемое значение

true в случае удачи, иначе false.

© 2000-2025, MetaQuotes Ltd.

Load

Читает данные дерева из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого для чтения бинарного файла.

Возвращаемое значение

true в случае удачи, иначе false.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа дерева.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа дерева.

© 2000-2025, MetaQuotes Ltd.

Шаблонные коллекции данных

Библиотека содержит классы и интерфейсы для определения шаблонных коллекций, которые, в свою очередь, дают пользователю возможность создавать строго типизированные коллекции. Они обеспечивают большее удобство и высокую производительность работы с данными, чем обычные типизированные коллекции.

Библиотека размещается в рабочем каталоге терминала в папке Include\Generic.

Объекты:

Объект	Описание	Тип
ICollection	Интерфейс для реализации шаблонных коллекций данных	INTERFACE
IEqualityComparable	Интерфейс для реализации объектов, которые можно сравнивать между собой	INTERFACE
IComparable	Интерфейс для реализации объектов, которые можно сравнивать между собой посредством отношения "больше, меньше или равно"	INTERFACE
IComparer	Интерфейс для реализации универсального класса, который сравнивает два объекта типа T на отношение "больше, меньше или равно"	INTERFACE
IEqualityComparer	Интерфейс для реализации универсального класса, который сравнивает два объекта типа T на равенство	INTERFACE
IList	Интерфейс для реализации шаблонных списков данных	INTERFACE
IMap	Интерфейс для реализации шаблонных коллекций пар "ключ — значение"	INTERFACE
ISet	Интерфейс для реализации шаблонных множеств данных	INTERFACE
CDefaultComparer	Вспомогательный класс, реализующий шаблонный интерфейс IComparer<T> на основе глобальных методов Compare	CLASS
CDefaultEqualityComparer	Вспомогательный класс, реализующий шаблонный интерфейс	CLASS

© 2000-2025, MetaQuotes Ltd.

Объект	Описание	Тип
	IEqualityComparer<T> глобальными методами Equals<T> и GetHashCode	
CArrayList	Шаблонный класс, реализующий интерфейс IList<T>	CLASS
CKeyValuePair	Класс реализует пару "ключ – значение"	CLASS
CHashMap	Шаблонный класс, реализующий интерфейс IMap<TKey, TValue>	CLASS
CHashSet	Шаблонный класс, реализующий интерфейс ISet<T>	CLASS
CLinkedListNode	Вспомогательный класс, необходимый для реализации класса CLinkedListNode<T>	CLASS
CLinkedList	Шаблонный класс, реализующий интерфейс ICollection<T>	CLASS
CQueue	Шаблонный класс, реализующий интерфейс ICollection<T>	CLASS
CRedBlackTreeNode	Вспомогательный класс, необходимый для реализации класса CRedBlackTree<T>	CLASS
CRedBlackTree	Шаблонный класс, реализующий интерфейс ICollection<T>	CLASS
CSortedMap	Шаблонный класс, реализующий интерфейс IMap<TKey, TValue>	CLASS
CSortedSet	Шаблонный класс, реализующий интерфейс ISet<T>	CLASS
CStack	Шаблонный класс, реализующий интерфейс ICollection<T>	CLASS

Глобальные методы:

Метод	Описание
ArrayBinarySearch	Ищет указанное значение в отсортированном по возрастанию одномерном массиве, используя интерфейс IComparable<T> для сравнения элементов
ArrayIndexOf	Ищет первое вхождение значения в одномерном массиве
ArrayLastIndexOf	Ищет последнее вхождение значения в одномерном массиве

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
ArrayReverse	Изменяет последовательность элементов в одномерном массиве
Compare	Сравнивает два значения на отношение "Больше, меньше или равно"
Equals	Сравнивает два значения на равенство
GetHashCode	Вычисляет значение хэш-кода

© 2000-2025, MetaQuotes Ltd.

ICollection<T>

ICollection<T> — интерфейс для реализации шаблонных коллекций данных.

Описание

Интерфейс ICollection<T> определяет основные методы для работы с коллекциями: подсчет количества элементов, очистку коллекции, добавление и удаление элементов и другие.

Декларация

```
template<typename T>
interface ICollection
```

Заголовок

```
#include <Generic\Interfaces\ICollection.mqh>
```

Иерархия наследования

ICollection

Прямые потомки

CLinkedList, CQueue, CRedBlackTree, CStack, IList, IMap, ISet

Методы класса

Метод	Описание
Add	Добавляет элемент в коллекцию
Count	Возвращает количество элементов в коллекции
Contains	Определяет, содержит ли коллекция элемент с указанным значением
CopyTo	Копирует все элементы коллекции в указанный массив, начиная с определенного индекса
Clear	Удаляет все элементы коллекции
Remove	Удаляет первое вхождение указанного элемента из коллекции

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в коллекцию.

```
bool Add(
   T  value     // значение элемента
   );
```

Параметры

value

[in] Значение элемента для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в коллекции.

```
int Count();
```

Возвращаемое значение

Возвращает количество элементов.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли коллекция элемент с указанным значением.

```
bool Contains(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает true, если в коллекции есть элемент с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все элементы коллекции в указанный массив, начиная с определенного индекса.

```
int CopyTo(
   T&          dst_array[],      // массив для записи
   const int   dst_start=0       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны элементы коллекции.

dst_start=0

[in] Индекс в массиве, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных элементов.

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы коллекции.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет первое вхождение указанного элемента из коллекции.

```
bool Remove(
   T  item     // значение элемента
   );
```

Параметры

item

[in] Значение элемента, которое нужно удалить.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

IEqualityComparable<T>

Интерфейс IEqualityComparable<T> — интерфейс для реализации объектов, которые можно сравнивать между собой.

Описание

Интерфейс IEqualityComparable<T> определяет методы получения хэш-кода текущего объекта и его сравнения на равенство с другим объектом того же типа.

Декларация

```
template<typename T>
interface IEqualityComparable
```

Заголовок

```
#include <Generic\Interfaces\IEqualityComparable.mqh>
```

Иерархия наследования

IEqualityComparable

Прямые потомки

IComparable

Методы класса

Метод	Описание
Equals	Сравнивает текущий объект с заданным значением
HashCode	Вычисляет значение хэш-кода для текущего объекта

© 2000-2025, MetaQuotes Ltd.

Equals

Сравнивает текущий объект с заданным значением.

```
bool Equals(
   T  value     // значение для сравнения
   );
```

Параметры

value

[in] Значение, с которым сравнивается текущий объект.

Возвращаемое значение

Возвращает true, если объекты равны, иначе false.

© 2000-2025, MetaQuotes Ltd.

HashCode

Вычисляет значение хэш-кода для текущего объекта.

```
int HashCode();
```

Возвращаемое значение

Возвращает хэш-код.

© 2000-2025, MetaQuotes Ltd.

IComparable<T>

Интерфейс IComparable<T> — интерфейс для реализации объектов, которые можно сравнивать между собой посредством отношения "больше, меньше или равно".

Описание

Интерфейс IComparable<T> определяет метод сравнения текущего объекта с другим того же типа, на основе которого коллекция этих объектов может быть отсортирована.

Декларация

```
template<typename T>
interface IComparable : public IEqualityComparable<T>
```

Заголовок

```
#include <Generic\Interfaces\IComparable.mqh>
```

Иерархия наследования

IEqualityComparable

IComparable

Прямые потомки

CKeyValuePair

Методы класса

Метод	Описание
Compare	Сравнивает текущий объект с указанным значением

© 2000-2025, MetaQuotes Ltd.

Compare

Сравнивает текущий объект с указанным значением.

```
int Compare(
   T  value     // значение для сравнения
   );
```

Параметры

value

[in] Значение, с которым сравнивается текущий объект.

Возвращаемое значение

Возвращает число, выражающее отношение текущего и переданного объекта:

- результат меньше нуля — текущий объект меньше переданного
- результат равен нулю — текущий объект равен переданному
- результат больше нуля — текущий объект больше переданного

© 2000-2025, MetaQuotes Ltd.

IComparer<T>

Интерфейс IComparer<T> — интерфейс для реализации универсального класса, который проводит сравнение двух объектов типа T между собой посредством отношения "Больше, меньше или равно".

Описание

Интерфейс IComparer<T> определяет метод сравнения двух объектов типа T, на основе которого коллекция этих объектов может быть отсортирована.

Декларация

```
template<typename T>
interface IComparer
```

Заголовок

```
#include <Generic\Interfaces\IComparer.mqh>
```

Иерархия наследования

IComparer

Прямые потомки

CDefaultComparer

Методы класса

Метод	Описание
Compare	Сравнивает два значения типа T

© 2000-2025, MetaQuotes Ltd.

Compare

Сравнивает два значения типа T.

```
int Compare(
   T  x,      // первое значение
   T  y       // второе значение
   );
```

Параметры

x

[in] Первое значение для сравнения.

y

[in] Первое значение для сравнения.

Возвращаемое значение

Возвращает число, выражающее отношение двух сравниваемых значений:

- результат меньше нуля — x меньше y (x<y)
- результат равен нулю — x равен y (x=y)
- результат больше нуля — x больше y (x>y)

© 2000-2025, MetaQuotes Ltd.

IEqualityComparer<T>

Интерфейс IEqualityComparer<T> — интерфейс для реализации универсального класса, который проводит сравнение двух объектов типа T.

Описание

Интерфейс IEqualityComparer<T> определяет методы получения хэш-кода объекта типа T и сравнения на равенство двух объектов типа T.

Декларация

```
template<typename T>
interface IEqualityComparer
```

Заголовок

```
#include <Generic\Interfaces\IEqualityComparer.mqh>
```

Иерархия наследования

IEqualityComparer

Прямые потомки

CDefaultEqualityComparer

Методы класса

Метод	Описание
Equals	Сравнивает два значения типа T
HashCode	Вычисляет значение хэш-кода от объекта типа T

© 2000-2025, MetaQuotes Ltd.

Equals

Сравнивает два значения типа T.

```
bool Equals(
   T  x,      // первое значение
   T  y       // второе значение
   );
```

Параметры

x

[in] Первое значение для сравнения.

y

[in] Второе значение для сравнения.

Возвращаемое значение

Возвращает true, если значения равны, иначе false.

© 2000-2025, MetaQuotes Ltd.

HashCode

Вычисляет значение хэш-кода от объекта типа T.

```
int HashCode(
   T  value     // объект для вычисления
   );
```

Параметры

value

[in] Объект, для которого нужно получить хэш-код.

Возвращаемое значение

Возвращает хэш-код.

© 2000-2025, MetaQuotes Ltd.

IList<T>

Интерфейс IList<T> — интерфейс для реализации шаблонных списков данных.

Описание

Интерфейс IList<T> определяет основные методы для работы со списками: доступ к элементу по индексу, поиск и удаление элемента, сортировку и другие.

Декларация

```
template<typename T>
interface IList : public ICollection<T>
```

Заголовок

```
#include <Generic\Interfaces\IList.mqh>
```

Иерархия наследования

ICollection

IList

Прямые потомки

CArrayList

Методы класса

Метод	Описание
TryGetValue	Получает элемент из списка по заданному индексу
TrySetValue	Изменяет значение из списка по заданному индексу
Insert	Вставляет элемент в список по указанному индексу
IndexOf	Ищет первое вхождение значения в списке
LastIndexOf	Ищет последнее вхождение значения в списке
RemoveAt	Удаляет элемент из списка по указанному индексу

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Получает элемент из списка по заданному индексу.

```
bool TryGetValue(
   const int  index,      // индекс элемента
   T&         value       // переменная для записи
   );
```

Параметры

index

[in] Индекс элемента из списка.

&value

[out] Переменная, в которую будет записано указанное значение элемента из списка.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Изменяет значение из списка по заданному индексу.

```
bool TrySetValue(
   const int  index,      // индекс элемента
   T          value       // новое значение
   );
```

Параметры

index

[in] Индекс элемента из списка.

value

[in] Новое значение, которое нужно присвоить указанному элементу списка.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в список по указанному индексу.

```
bool Insert(
   const int  index,      // индекс для вставки
   T          item        // значение для вставки
   );
```

Параметры

index

[in] Индекс для вставки.

item

[in] Значение, которое необходимо вставить по указанному индексу.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

IndexOf

Ищет первое вхождение значения в списке.

```
int IndexOf(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает индекс первого найденного элемента. Если значение не найдено, возвращает -1.

© 2000-2025, MetaQuotes Ltd.

LastIndexOf

Ищет последнее вхождение значения в списке.

```
int LastIndexOf(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает индекс последнего найденного элемента. Если значение не найдено, возвращает -1.

© 2000-2025, MetaQuotes Ltd.

RemoveAt

Удаляет элемент из списка по указанному индексу.

```
bool RemoveAt(
   const int  index     // индекс элемента
   );
```

Параметры

index

[in] Индекс элемента, который необходимо удалить.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

IMap<TKey, TValue>

Интерфейс IMap<TKey, TValue> — интерфейс для реализации шаблонных коллекций пар "ключ — значение".

Описание

Интерфейс IMap<TKey, TValue> определяет основные методы для работы с коллекциями, данные в которых хранятся в виде пар "ключ — значение".

Декларация

```
template<typename TKey, typename TValue>
interface IMap : public ICollection<TKey>
```

Заголовок

```
#include <Generic\Interfaces\IMap.mqh>
```

Иерархия наследования

ICollection

IMap

Прямые потомки

CHashMap, CSortedMap

Методы класса

Метод	Описание
Add	Добавляет пару "ключ — значение" в коллекцию
Contains	Определяет, содержит ли коллекция пару "ключ — значение" с указанным ключом и значением
Remove	Удаляет первое вхождение пары "ключ — значение" с указанным ключом из коллекции
TryGetValue	Получает элемент из коллекции по заданному ключу
TrySetValue	Изменяет значение пары "ключ — значение" из коллекции по заданному ключу
CopyTo	Копирует все пары "ключ — значение" из коллекции в указанные массивы, начиная с определенного индекса

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет пару "ключ — значение" в коллекцию.

```
bool Add(
   TKey    key,       // ключ
   TValue  value      // значение
   );
```

Параметры

key

[in] Ключ.

value

[in] Значение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли коллекция пару "ключ — значение" с указанным ключом и значением.

```
bool Contains(
   TKey    key,       // ключ
   TValue  value      // значение
   );
```

Параметры

key

[in] Ключ.

value

[in] Значение.

Возвращаемое значение

Возвращает true, если в коллекции есть пара "ключ — значение" с указанными ключом и значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет первое вхождение пары "ключ — значение" с указанным ключом из коллекции.

```
bool Remove(
   TKey  key      // ключ
   );
```

Параметры

key

[in] Ключ.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Получает элемент из коллекции по заданному ключу.

```
bool TryGetValue(
   TKey      key,       // ключ
   TValue&   value      // переменная для записи значения
   );
```

Параметры

key

[in] Ключ.

&value

[out] Переменная, в которую будет записано указанное значение пары "ключ — значение".

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Изменяет значение пары "ключ — значение" из коллекции по заданному ключу.

```
bool TrySetValue(
   TKey    key,       // ключ
   TValue  value      // новое значение
   );
```

Параметры

key

[in] Ключ.

value

[in] Новое значение, которое нужно присвоить указанной паре "ключ-значение".

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все пары "ключ — значение" из коллекции в указанные массивы, начиная с определенного индекса.

```
int CopyTo(
   TKey&       dst_keys[],       // массив для записи ключей
   TValue&     dst_values[],     // массив для записи значений
   const int   dst_start=0       // начальный индекс для записи
   );
```

Параметры

&dst_keys[]

[out] Массив, в который будут записаны все ключи из коллекции.

&dst_values[]

[out] Массив, в который будут записаны значения соответствующих ключей из коллекции.

dst_start=0

[in] Индекс в массивах, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных пар "ключ — значение".

© 2000-2025, MetaQuotes Ltd.

ISet<T>

Интерфейс ISet<T> — интерфейс для реализации шаблонных множеств данных.

Описание

Интерфейс ISet<T> определяет основные методы для работы с множествами: объединение и пересечение множеств, определение строгих и нестрогих подмножеств и другие.

Декларация

```
template<typename T>
interface ISet : public ICollection<T>
```

Заголовок

```
#include <Generic\Interfaces\ISet.mqh>
```

Иерархия наследования

ICollection

ISet

Прямые потомки

CHashSet, CSortedSet

Методы класса

Метод	Описание
ExceptWith	Выполняет операцию разности текущей и переданной коллекции (массива)
IntersectWith	Выполняет операцию пересечения текущей и переданной коллекции (массива)
SymmetricExceptWith	Выполняет операцию симметричной разности текущей и переданной коллекции (массива)
UnionWith	Выполняет операцию объединения текущей и переданной коллекции (массива)
IsProperSubsetOf	Определяет, является ли текущее множество строгим подмножеством заданной коллекции или массива
IsProperSupersetOf	Определяет, является ли текущее множество строгим надмножеством заданной коллекции или массива
IsSubsetOf	Определяет, является ли текущее множество подмножеством заданной коллекции или массива
IsSupersetOf	Определяет, является ли текущее множество надмножеством заданной коллекции или массива

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
Overlaps	Определяет, пересекается ли текущее множество с заданной коллекцией или массивом
SetEquals	Определяет, содержит ли текущее множество все элементы из заданной коллекции или массива

© 2000-2025, MetaQuotes Ltd.

ExceptWith

Выполняет операцию разности текущей и переданной коллекции (массива). То есть удаляет из текущей коллекции (массива) все элементы, которые есть также и в указанной коллекции (массиве).

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void ExceptWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, которая будет вычитаться из текущего множества.

&collection[]

[in] Массив, который будет вычитаться из текущего множества.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

IntersectWith

Выполняет операцию пересечения текущей и переданной коллекции (массива). То есть оставляет в текущей коллекции только те элементы, которые имеются также и в указанной коллекции (массиве).

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void IntersectWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, с которой будет строиться произведение.

&collection[]

[in] Массив, с которым будет строиться произведение.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

Выполняет операцию симметричной разности текущей и переданной коллекции (массива). То есть в текущей коллекции (массиве) будут содержатся только те элементы, которые имелись либо в исходном объекте, либо в переданном, но не одновременно в них обоих.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void SymmetricExceptWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, с которой будет строиться симметричная разность.

&collection[]

[in] Массив, с которым будет строиться симметричная разность.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

UnionWith

Выполняет операцию объединения текущей и переданной коллекции (массива). То есть добавляет в текущую коллекцию (массив) отсутствующие элементы из указанной коллекции (массива).

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void UnionWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, с которой будет строиться сумма.

&collection[]

[in] Массив, с которым будет строиться сумма.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

Определяет, является ли текущее множество строгим подмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsProperSubsetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true в случае, если текущее множество является строгим подмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

Определяет, является ли текущее множество строгим надмножеством заданной коллекции или массива.

Версия для работы с коллекцией реализующей интерфейс ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsProperSupersetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true, если текущее множество является строгим надмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsSubsetOf

Определяет, является ли текущее множество подмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsSubsetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true, если текущее множество является подмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

Определяет, является ли текущее множество надмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsSupersetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true в случае, если текущее множество является надмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

Overlaps

Определяет, пересекается ли текущее множество с заданной коллекцией или массивом.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // коллекция для сравнения
   );
```

Версия для работы с массивом.

```
bool Overlaps(
   T&  array[]                     // массив для сравнения
   );
```

Параметры

**collection*

[in] Коллекция для определения пересечения.

&collection[]

[in] Массив для определения пересечения.

Возвращаемое значение

Возвращает true в случае, если между текущим множеством и коллекцией или массивом есть пересечение, иначе false.

© 2000-2025, MetaQuotes Ltd.

SetEquals

Определяет, содержит ли текущее множество все элементы из заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // коллекция для сравнения
   );
```

Версия для работы с массивом.

```
bool SetEquals(
   T&  array[]                     // массив для сравнения
   );
```

Параметры

**collection*

[in] Коллекция для сопоставления элементов.

&collection[]

[in] Коллекция для сопоставления элементов.

Возвращаемое значение

Возвращает true в случае, если в текущем множестве есть все элементы указанной коллекции или массива, иначе false.

© 2000-2025, MetaQuotes Ltd.

CDefaultComparer<T>

Класс CDefaultComparer<T> — вспомогательный класс, который реализует шаблонный интерфейс IComparer<T> на основе глобальных методов Compare.

Описание

Класс CDefaultComparer<T> используется по умолчанию в шаблонных коллекциях данных, если пользователь явно не использует другой класс, реализующий интерфейс IComparer<T>.

Декларация

```
template<typename T>
class CDefaultComparer : public IComparer<T>
```

Заголовок

```
#include <Generic\Internal\DefaultComparer.mqh>
```

Иерархия наследования

IComparer

CDefaultComparer

Методы класса

Метод	Описание
Compare	Сравнивает два значения типа T

© 2000-2025, MetaQuotes Ltd.

Compare

Сравнивает два значения типа T.

```
int Compare(
   T  x,      // первое значение
   T  y       // второе значение
   );
```

Параметры

x

[in] Первое значение для сравнения.

y

[in] Второе значение для сравнения.

Возвращаемое значение

Возвращает число, выражающее отношение двух сравниваемых значений:

- результат меньше нуля — x меньше y (x<y)
- результат равен нулю — x равен y (x=y)
- результат больше нуля — x больше y (x>y)

Примечание

Сравнение значений x и y происходит на основе одной из перегрузок глобального метода Compare, в зависимости от типа T.

© 2000-2025, MetaQuotes Ltd.

CDefaultEqualityComparer<T>

Класс CDefaultEqualityComparer<T> — вспомогательный класс, который реализует шаблонный интерфейс IEqualityComparer<T> на основе глобальных методов Equals<T> и GetHashCode.

Описание

Класс CDefaultEqualityComparer<T> используется по умолчанию в шаблонных коллекциях данных, если пользователь явно не использует другой класс, реализующий интерфейс IEqualityComparer<T>.

Декларация

```
template<typename T>
class CDefaultEqualityComparer : public IEqualityComparer<T>
```

Заголовок

```
#include <Generic\Internal\DefaultEqualityComparer.mqh>
```

Иерархия наследования

IEqualityComparer

CDefaultEqualityComparer

Методы класса

Метод	Описание
Equals	Сравнивает два значения типа T
HashCode	Вычисляет значение хэш-кода от объекта типа T

© 2000-2025, MetaQuotes Ltd.

Equals

Сравнивает два значения типа T.

```
bool Equals(
   T  x,      // первое значение
   T  y       // второе значение
   );
```

Параметры

x

[in] Первое значение для сравнения.

y

[in] Второе значение для сравнения.

Возвращаемое значение

Возвращает true в если значения равны, иначе false.

© 2000-2025, MetaQuotes Ltd.

HashCode

Вычисляет значение хэш-кода от объекта типа T.

```
int HashCode(
   T  value     // объект для вычисления
   );
```

Параметры

value

[in] Объект для которого нужно получить хэш-код.

Возвращаемое значение

Возвращает хэш-код.

© 2000-2025, MetaQuotes Ltd.

CRedBlackTreeNode<T>

Класс CRedBlackTreeNode<T> — вспомогательный класс, необходимый для реализации класса CRedBlackTree<T>.

Описание

Класс CRedBlackTreeNode<T> — узел красно-черного дерева CRedBlackTree<T>. В этом классе реализованы методы для навигации по дереву.

Декларация

```
template<typename T>
class CRedBlackTreeNode
```

Заголовок

```
#include <Generic\RedBlackTree.mqh>
```

Методы класса

Метод	Описание
Value	Возвращает и устанавливает значение узла
Parent	Возвращает и устанавливает указатель на родительский узел
Left	Возвращает и устанавливает указатель на левый узел
Right	Возвращает и устанавливает указатель на правый узел
Color	Возвращает и устанавливает цвет узла
IsLeaf	Определяет, является ли данный узел листом
CreateEmptyNode	Создаёт новый узел черного цвета без предка и потомков и возвращает указатель на него

© 2000-2025, MetaQuotes Ltd.

Value (метод Get)

Возвращает значение узла.

```
T Value();
```

Возвращаемое значение

Возвращает значение узла.

Value (метод Set)

Устанавливает значение узла.

```
void Value(
   T  value     // значение узла
   );
```

Параметры

value

[in] Значение узла.

© 2000-2025, MetaQuotes Ltd.

Parent (метод Get)

Возвращает указатель на родительский узел.

```
CRedBlackTreeNode<T>* Parent();
```

Возвращаемое значение

Возвращает указатель на родительский узел.

Parent (метод Set)

Устанавливает указатель на родительский узел.

```
void Parent(
   CRedBlackTreeNode<T>*  node     // указатель на родительский узел
   );
```

Параметры

node

[in] Указатель на родительский узел.

© 2000-2025, MetaQuotes Ltd.

Left (метод Get)

Возвращает указатель на левый узел.

```
CRedBlackTreeNode<T>* Left();
```

Возвращаемое значение

Возвращает указатель на левый узел.

Left (метод Set)

Устанавливает указатель на левый узел.

```
void Left(
   CRedBlackTreeNode<T>*  node     // указатель на левый узел
   );
```

Параметры

node

[in] Указатель на левый узел.

© 2000-2025, MetaQuotes Ltd.

Right (метод Get)

Возвращает указатель на правый узел.

```
CRedBlackTreeNode<T>* Right();
```

Возвращаемое значение

Возвращает указатель на правый узел.

Right (метод Set)

Устанавливает указатель на правый узел.

```
void Right(
   CRedBlackTreeNode<T>*  node     // указатель на правый узел
   );
```

Параметры

node

[in] Указатель на правый узел.

© 2000-2025, MetaQuotes Ltd.

Color (метод Get)

Возвращает цвет узла.

```
ENUM_RED_BLACK_TREE_NODE_TYPE Color();
```

Возвращаемое значение

Возвращает цвет узла.

Color (метод Set)

Устанавливает цвет узла.

```
void Color(
   ENUM_RED_BLACK_TREE_NODE_TYPE  clr     // цвет узла
   );
```

Параметры

clr

[in] Цвет узла.

Примечание

Цвет узла задаётся значением из перечисления ENUM_RED_BLACK_TREE_NODE_TYPE и бывает двух типов:

- RED_BLACK_TREE_NODE_RED — красный цвет узла;
- RED_BLACK_TREE_NODE_BLACK — черный цвет узла.

© 2000-2025, MetaQuotes Ltd.

IsLeaf

Определяет, является ли данный узел листом.

```
bool IsLeaf();
```

Возвращаемое значение

Возвращает true, если узел является листом, иначе false.

© 2000-2025, MetaQuotes Ltd.

CreateEmptyNode

Создаёт новый узел черного цвета без предка и потомков и возвращает указатель на него.

```
static CRedBlackTreeNode<T>* CreateEmptyNode();
```

Возвращаемое значение

Возвращает указатель на новый узел.

© 2000-2025, MetaQuotes Ltd.

CLinkedListNode<T>

Класс CLinkedListNode<T> — вспомогательный класс, необходимый для реализации класса CLinkedListNode<T>.

Описание

Класс CLinkedListNode<T> является узлом двунаправленного списка CLinkedListNode<T>. В классе реализованы методы для навигации по списку.

Декларация

```
template<typename T>
class CLinkedListNode
```

Заголовок

```
#include <Generic\LinkedList.mqh>
```

Методы класса

Метод	Описание
List	Возвращает и устанавливает указатель на двунаправленный список CLinkedList<T>
Next	Возвращает и устанавливает указатель на следующий узел
Previous	Возвращает и устанавливает указатель на предыдущий узел
Value	Возвращает и устанавливает значение узла

© 2000-2025, MetaQuotes Ltd.

List (метод Get)

Возвращает указатель на двунаправленный список CLinkedList<T>.

```
CLinkedList<T>* List();
```

Возвращаемое значение

Возвращает указатель на двунаправленный список CLinkedList<T>.

List (метод Set)

Устанавливает указатель на двунаправленный список CLinkedList<T>.

```
void List(
   CLinkedList<T>*  value     // указатель на список
   );
```

Параметры

value

[in] Указатель на двунаправленный список CLinkedList<T>.

© 2000-2025, MetaQuotes Ltd.

Next (метод Get)

Возвращает указатель на следующий узел.

```
CLinkedListNode<T>* Next();
```

Возвращаемое значение

Возвращает указатель на следующий узел.

Next (метод Set)

Устанавливает указатель на следующий узел.

```
void Next(
   CLinkedListNode<T>*  value     // указатель на следующий узел
   );
```

Параметры

value

[in] Указатель на следующий узел.

© 2000-2025, MetaQuotes Ltd.

Previous (метод Get)

Возвращает указатель на предыдущий узел.

```
CLinkedListNode<T>* Previous();
```

Возвращаемое значение

Возвращает указатель на предыдущий узел.

Previous (метод Set)

Устанавливает указатель на предыдущий узел.

```
void Previous(
   CLinkedListNode<T>*  value     // указатель на предыдущий узел
   );
```

Параметры

value

[in] Указатель на предыдущий узел.

© 2000-2025, MetaQuotes Ltd.

Value (метод Get)

Возвращает значение узла.

```
T Value();
```

Возвращаемое значение

Возвращает значение узла.

Value (метод Set)

Устанавливает значение узла.

```
void Value(
   T  value     // значение узла
   );
```

Параметры

value

[in] Значение узла.

© 2000-2025, MetaQuotes Ltd.

CKeyValuePair<TKey,TValue>

Класс CKeyValuePair<TKey,TValue> — класс реализует пару "ключ — значение".

Описание

Класс CKeyValuePair<TKey,TValue> реализует методы непосредственной работы с ключом и значением пары "ключ — значение".

Декларация

```
template<typename TKey, typename TValue>
class CKeyValuePair : public IComparable<CKeyValuePair<TKey,TValue>*>
```

Заголовок

```
#include <Generic\HashMap.mqh>
```

Иерархия наследования

IEqualityComparable

IComparable

CKeyValuePair

Методы класса

Метод	Описание
Key	Возвращает и устанавливает ключ пары "ключ — значение"
Value	Возвращает и устанавливает значение пары "ключ — значение"
Clone	Создаёт новую пару "ключ — значение", ключ и значение которой равны текущим
Compare	Сравнивает текущую пару "ключ — значение" с указанной
Equals	Сравнивает текущую пару "ключ — значение" с указанной на предмет равенства
HashCode	Вычисляет значение хэш-кода от пары "ключ — значение"

© 2000-2025, MetaQuotes Ltd.

Key (метод Get)

Возвращает ключ пары "ключ — значение".

```
TKey Key();
```

Возвращаемое значение

Возвращает ключ.

Key (метод Set)

Устанавливает значение ключа пары "ключ — значение".

```
void Key(
   TKey  key      // ключ
   );
```

Параметры

key

[in] Ключ.

© 2000-2025, MetaQuotes Ltd.

Value (метод Get)

Возвращает значение пары "ключ — значение".

```
TValue Value();
```

Возвращаемое значение

Возвращает значение.

Value (метод Set)

Устанавливает значение пары "ключ — значение".

```
void Value(
   TValue  value     // значение
   );
```

Параметры

value

[in] Значение

© 2000-2025, MetaQuotes Ltd.

Clone

Создаёт новую пару "ключ — значение", ключ и значение которой равны текущим.

```
TValue>* Clone();
```

Возвращаемое значение

Вовращает новую пару "ключ — значение"

© 2000-2025, MetaQuotes Ltd.

Compare

Сравнивает текущую пару "ключ — значение" с указанной.

```
int Compare(
   CKeyValuePair<TKeyTValue>*  pair     // пара для сравнения
   );
```

Параметры

pair

[in] Пара для сравнения.

Возвращаемое значение

Возвращает число, выражающее отношение текущей и переданной пары "ключ — значение":

- результат меньше нуля — текущая пара "ключ-значение" меньше переданной
- результат равен нулю — текущая пара "ключ-значение" равна переданной
- результат больше нуля — текущая пара "ключ-значение" больше переданной

Примечание

Сравнение двух пар "ключ — значение" происходит по ключу.

© 2000-2025, MetaQuotes Ltd.

Equals

Сравнивает текущую пару "ключ — значение" с указанной на предмет равенства.

```
bool Equals(
   CKeyValuePair<TKeyTValue>*  pair     // пара для сравнения
   );
```

Параметры

pair

[in] Пара для сравнения

Возвращаемое значение

Возвращает true, если пары "ключ — значение" равны, иначе false.

Примечание

Сравнение двух пар "ключ — значение" происходит по ключу.

© 2000-2025, MetaQuotes Ltd.

HashCode

Вычисляет значение хэш-кода от пары "ключ — значение".

```
int HashCode();
```

Возвращаемое значение

Возвращает хэш-код.

Примечание

Хэш-код пары "ключ — значение" равен хэш-коду ключа.

© 2000-2025, MetaQuotes Ltd.

CArrayList<T>

Класс CArrayList<T> — шаблонный класс, реализующий интерфейс IList<T>.

Описание

Класс CArrayList<T> является реализацией динамического списка данных типа T. Данный класс предоставляет основные методы для работы со списком: доступ к элементу по индексу, поиск и удаление элемента, сортировку и другие.

Декларация

```
template<typename T>
class CArrayList : public IList<T>
```

Заголовок

```
#include <Generic\ArrayList.mqh>
```

Иерархия наследования

ICollection
  IList
    CArrayList

Методы класса

Метод	Описание
Capacity	Возвращает и устанавливает текущую емкость списка
Count	Возвращает количество элементов в списке
Contains	Определяет, содержит ли список элемент с указанным значением
TrimExcess	Сокращает емкость списка до фактического числа элементов
TryGetValue	Получает элемент списка по указанному индексу
TrySetValue	Устанавливает значение элемента списка по указанному индексу
Add	Добавляет элемент в список
AddRange	Добавляет коллекцию или массив элементов в список
Insert	Вставляет элемент в список по указанному индексу
InsertRange	Вставляет коллекцию или массив элементов в список по указанному индексу
CopyTo	Копирует все элементы списка в указанный массив, начиная с определенного индекса
BinarySearch	Ищет указанное значение в отсортированном по возрастанию списке
IndexOf	Ищет первое вхождение значения в списке

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
LastIndexOf	Ищет последнее вхождение значения в списке
Clear	Удаляет все элементы коллекции
Remove	Удаляет первое вхождение указанного элемента из списка
RemoveAt	Удаляет указанный элемент из списка по заданному индексу
RemoveRange	Удаляет диапазон элементов из списка
Reverse	Изменяет последовательность элементов в списке
Sort	Выполняет сортировку элементов списка

© 2000-2025, MetaQuotes Ltd.

Capacity (метод Get)

Возвращает текущую емкость списка.

```
int Capacity();
```

Возвращаемое значение

Возвращает текущую емкость списка.

Capacity (метод Set)

Устанавливает текущую емкость списка.

```
void Capacity(
   const int  capacity     // значение емкости
   );
```

Параметры

capacity

[in] Новое значение емкости.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в списке.

```
int Count();
```

Возвращаемое значение

Возвращает количество элементов.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли список элемент с указанным значением.

```
bool Contains(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает true, если в списке есть элемент с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

Сокращает емкость списка до фактического числа элементов, тем самым освобождая неиспользованную память.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Получает элемент списка по указанному индексу.

```
bool TryGetValue(
   const int  index,      // индекс
   T&         value       // переменная для записи
   );
```

Параметры

index

[in] Индекс элемента списка, значение которого необходимо получить.

&value

[out] Переменная для записи значения элемента.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Устанавливает значение элемента списка по указанному индексу.

```
bool TrySetValue(
   const int  index,      // индекс
   T          value       // значение элемента
   );
```

Параметры

index

[in] Индекс элемента списка, значение которого необходимо установить.

value

[in] Устанавливаемое значение элемента списка.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в список.

```
bool Add(
   T  value     // значение элемента
   );
```

Параметры

value

[in] Значение элемента для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

AddRange

Добавляет коллекцию или массив элементов в список.

Версия для добавления массива.

```
bool AddRange(
   const T&  array[]               // массив для добавления
   );
```

Версия для добавления коллекции.

```
bool AddRange(
   ICollection<T>*  collection     // коллекция для добавления
   );
```

Параметры

&array[]

[in] Массив для добавления.

**collection*

[in] Коллекция для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет элемент в список по указанному индексу.

```
bool Insert(
   const int  index,      // индекс для вставки
   T          item        // значение для встаки
   );
```

Параметры

index

[in] Индекс для вставки.

item

[in] Значение, которое необходимо вставить по указанному индексу.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

InsertRange

Вставляет коллекцию или массив элементов в список по указанному индексу.

Версия для вставки массива.

```
bool InsertRange(
   const int  index,                // индекс для вставки
   const T&   array[]               // массив для вставки
   );
```

Версия для вставки коллекции.

```
bool InsertRange(
   const int         index,         // индекс для вставки
   ICollection<T>*   collection     // коллекция для вставки
   );
```

Параметры

index

[in] Индекс для вставки.

&array[]

[in] Массив, который необходимо вставить по указанному индексу.

**collection*

[in] Коллекция, которую необходимо вставить по указанному индексу.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все элементы списка в указанный массив, начиная с определенного индекса.

```
int CopyTo(
   T&          dst_array[],      // массив для записи
   const int   dst_start=0       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны элементы списка.

dst_start=0

[in] Индекс в массиве, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных элементов.

© 2000-2025, MetaQuotes Ltd.

BinarySearch

Ищет указанное значение в отсортированном по возрастанию списке.

Версия для поиска по заданному диапазону значений и с использованием класса, реализующего интерфейс IComparable<T> для сравнения элементов.

```
int BinarySearch(
   const int       index,        // начальный индекс
   const int       count,        // диапазон поиска
   T               item,         // искомое значение
   IComparer<T>*   comparer      // интерфейс для сравнения
   );
```

Версия для поиска с использованием класса, реализующего интерфейс IComparable<T> для сравнения элементов.

```
int BinarySearch(
   T               item,         // искомое значение
   IComparer<T>*   comparer      // интерфейс для сравнения
   );
```

Версия для поиска с использованием глобального метода ::Compare для сравнения элементов.

```
int BinarySearch(
   T  item                       // искомое значение
   );
```

Параметры

index

[in] Начальный индекс, с которого начинается поиск.

count

[in] Длина диапазона поиска.

item

[in] Искомое значение.

**comparer*

[in] Интерфейс для сравнения элементов.

Возвращаемое значение

Возвращает индекс найденного элемента. Если искомое значение не найдено, то возвращает индекс меньшего элемента, которое ближе всех по значению.

© 2000-2025, MetaQuotes Ltd.

IndexOf

Ищет первое вхождение значения в списке.

Версия для поиска по всему списку.

```
int IndexOf(
   T  item                    // искомое значение
   );
```

Версия для поиска с указанной позиции и до конца списка.

```
int IndexOf(
   T          item,           // искомое значение
   const int  start_index     // начальный индекс
   );
```

Версия для поиска с указанной позиции в заданном диапазоне.

```
int IndexOf(
   T          item,           // искомое значение
   const int  start_index,    // начальный индекс
   const int  count           // диапазон поиска
   );
```

Параметры

item

[in] Искомое значение.

start_index

[in] Начальный индекс, с которого начинается поиск.

count

[in] Длина диапазона поиска.

Возвращаемое значение

Возвращает индекс первого найденного элемента. Если значение не найдено, возвращает -1.

© 2000-2025, MetaQuotes Ltd.

LastIndexOf

Ищет последнее вхождение значения в списке.

Версия для поиска по всему списку.

```
int LastIndexOf(
   T  item                    // искомое значение
   );
```

Версия для поиска с указанной позиции и до конца списка.

```
int LastIndexOf(
   T           item,          // искомое значение
   const int   start_index    // начальный индекс
   );
```

Версия для поиска с указанной позиции в заданном диапазоне.

```
int LastIndexOf(
   T           item,          // искомое значение
   const int   start_index,   // начальный индекс
   const int   count          // диапазон поиска
   );
```

Параметры

item

[in] Искомое значение.

start_index

[in] Начальный индекс, с которого начинается поиск.

count

[in] Длина диапазона поиска.

Возвращаемое значение

Возвращает индекс последнего найденного элемента. Если значение не найдено, возвращает -1.

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы коллекции.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет первое вхождение указанного элемента из списка.

```
bool Remove(
   T  item     // значение элемента
   );
```

Параметры

item

[in] Значение элемента, которое нужно удалить.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

RemoveAt

Удаляет указанный элемент из списка по заданному индексу.

```
bool RemoveAt(
   const int  index     // индекс
   );
```

Параметры

index

[in] Индекс удаляемого элемента.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

RemoveRange

Удаляет диапазон элементов из списка.

```
bool RemoveRange(
   const int  start_index,      // начальный индекс
   const int  count             // количество элементов
   );
```

Параметры

start_index

[in] Начальный индекс, с которого начинается удаление.

count

[in] Количество элементов, подлежащих удалению.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Reverse

Изменяет последовательность элементов в списке.

Версия для работы со всем списком.

```
bool Reverse();
```

Версия для работы с заданным диапазоном элементов списка.

```
bool Reverse(
   const int  start_index,     // начальный индекс
   const int  count            // количество элементов
   );
```

Параметры

start_index

[in] Начальный индекс.

count

[in] Количество элеменов списка, участвующих в операции.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Sort

Выполняет сортировку элементов списка.

Версия для сортировки всех элементов списка.

```
bool Sort();
```

Версия для сортировки всех элементов списка с использованием класса, реализующего интерфейс IComparable<T> для сравнения элементов.

```
bool Sort(
   IComparer<T>*  comparer          // интерфейс для сравнения
   );
```

Версия для сортировки заданного диапазона элементов списка с использованием класса, реализующего интерфейс IComparable<T> для сравнения элементов.

```
bool Sort(
   const int      start_index,      // начальный индекс
   const int      count,            // количество элементов
   IComparer<T>*  comparer          // интерфейс для сравнения
   );
```

Параметры

**comparer*

[in] Интерфейс для сравнения элементов.

start_index

[in] Начальный индекс, с которого начинается сортировка.

count

[in] Длина диапазона сортировки.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

CHashMap<TKey,TValue>

Класс CHashMap<TKey,TValue> — шаблонный класс, реализующий интерфейс IMap<TKey,TValue>.

Описание

Класс CHashMap<TKey,TValue> является реализацией динамической хэш-таблицы, данные которой хранятся в виде неупорядоченных пар "ключ — значение" с соблюдением требования уникальности ключа. Данный класс предоставляет основные методы для работы с хэш-таблицей: доступ к значению по ключу, поиск и удаление пары "ключ-значение" и другие.

Декларация

```
template<typename TKey, typename TValue>
class CHashMap : public IMap<TKey,TValue>
```

Заголовок

```
#include <Generic\HashMap.mqh>
```

Иерархия наследования

ICollection
  IMap
    CHashMap

Методы класса

Метод	Описание
Add	Добавляет пару "ключ — значение" в хэш-таблицу
Count	Возвращает количество элементов в хэш-таблице
Comparer	Возвращает указатель на интерфейс IEqualityComparer<T>, использующийся для организации хэш-таблицы
Contains	Определяет, содержит ли хэш-таблица указанную пару "ключ — значение"
ContainsKey	Определяет, содержит ли хэш-таблица пару "ключ — значение" с указанным ключом
ContainsValue	Класс CHashMap<TKey, TValue> — шаблонный класс, реализующий интерфейс IMap<TKey,TValue>
CopyTo	Копирует все пары "ключ — значение" из хэш-таблицы в указанные массивы, начиная с определенного индекса
Clear	Удаляет все элементы хэш-таблицы
Remove	Удаляет первое вхождение пары "ключ — значение" хэш-таблицы
TryGetValue	Получает элемент из хэш-таблицы по заданному ключу

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
TrySetValue	Изменяет значение пары "ключ — значение" из хэш-таблицы по заданному ключу

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет пару "ключ — значение" в хэш-таблицу.

Версия для добавления сформированной пары "ключ — значение".

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // пара "ключ — значение"
   );
```

Версия для добавления новой пары "ключ — значение" с указанным ключом и значением.

```
bool Add(
   TKey    key,                         // ключ
   TValue  value                        // значение
   );
```

Параметры

**pair*

[in] Пара "ключ — значение".

key

[in] Ключ.

value

[in] Значение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в хэш-таблице.

```
int Count();
```

© 2000-2025, MetaQuotes Ltd.

Comparer

Возвращает указатель на интерфейс IEqualityComparer<T>, использующийся для организации хэш-таблицы.

```
IEqualityComparer<TKey>* Comparer() const;
```

Возвращаемое значение

Возвращает указатель на интерфейс IEqualityComparer<T>.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли хэш-таблица указанную пару "ключ — значение".

Версия для работы с сформированной парой "ключ — значение".

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // пара "ключ — значение"
   );
```

Версия для работы с парой "ключ — значение" в виде отдельно заданного ключа и значения.

```
bool Contains(
   TKey   key,                          // ключ
   TValue value                         // значение
   );
```

Параметры

**item*

[in] Пара "ключ — значение".

key

[in] Ключ.

value

[in] Значение.

Возвращаемое значение

Возвращает true, если в хэш-таблице есть пара "ключ-значение" с указанным ключом и значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

ContainsKey

Определяет, содержит ли хэш-таблица пару "ключ — значение" с указанным ключом.

```
bool ContainsKey(
   TKey  key      // ключ
   );
```

Параметры

key

[in] Ключ.

Возвращаемое значение

Возвращает true если в хэш-таблице есть пара "ключ — значение" с указанным ключом, иначе false.

© 2000-2025, MetaQuotes Ltd.

ContainsValue

Определяет, содержит ли хэш-таблица пару "ключ — значение" с указанным значением.

```
bool ContainsValue(
   TValue  value     // значение
   );
```

Параметры

value

[in] Значение.

Возвращаемое значение

Возвращает true, если в хэш-таблице есть пара "ключ — значение" с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все пары "ключ — значение" из хэш-таблицы в указанные массивы, начиная с определенного индекса.

Версия для копирования хэш-таблицы в массив пар "ключ — значение".

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // массив для записи пар "ключ — знач
   const int                    dst_start=0      // начальный индекс для записи
   );
```

Версия для копирования хэш-таблицы в отдельные массивы для ключей и значений.

```
int CopyTo(
   TKey&       dst_keys[],                       // массив для записи ключей
   TValue&     dst_values[],                     // массив для записи значений
   const int   dst_start=0                       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны все пары из хэш-таблицы.

&dst_keys[]

[out] Массив, в который будут записаны все ключи из хэш-таблицы.

&dst_values[]

[out] Массив, в который будут записаны все значения из хэш-таблицы.

dst_start=0

[in] Индекс массива, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных пар "ключ — значение".

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы хэш-таблицы.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет первое вхождение пары "ключ — значение" хэш-таблицы.

Версия для удаления пары "ключ — значение" по сформированной паре "ключ — значение".

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // пара "ключ-значение"
   );
```

Версия для удаления пары "ключ — значение" по ключу.

```
bool Remove(
   TKey  key                           // ключ
   );
```

Параметры

**item*

[in] Пара "ключ — значение".

key

[in] Ключ.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Получает элемент из хэш-таблицы по заданному ключу.

```
bool TryGetValue(
   TKey      key,       // ключ
   TValue&   value      // переменная для записи значения
   );
```

Параметры

key

[in] Ключ.

&value

[out] Переменная, в которую будет записано указанное значение пары "ключ — значение".

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Изменяет значение пары "ключ — значение" из хэш-таблицы по заданному ключу.

```
bool TrySetValue(
   TKey    key,       // ключ
   TValue  value      // новое значение
   );
```

Параметры

key

[in] Ключ.

value

[in] Новое значение, которое нужно присвоить указанной паре "ключ — значение".

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

CHashSet<T>

Класс CHashSet<T> — шаблонный класс, реализующий интерфейс ISet<T>.

Описание

Класс CHashSet<T> является реализацией неупорядоченного динамического множества данных типа T, с соблюдением требования уникальности каждого значения. Данный класс предоставляет основные методы для работы с множествами и операциями над ними: объединением и пересечением множеств, определением строгих и нестрогих подмножеств и другие.

Декларация

```
template<typename T>
class CHashSet : public ISet<T>
```

Заголовок

```
#include <Generic\HashSet.mqh>
```

Иерархия наследования

ICollection
  ISet
    CHashSet

Методы класса

Метод	Описание
Add	Добавляет элемент в множество
Count	Возвращает количество элементов в множестве
Comparer	Определяет, содержит ли множество элемент с указанным значением
Contains	Возвращает указатель на интерфейс IEqualityComparer<T>, использующийся для организации множества
TrimExcess	Сокращает емкость множества до фактического числа элементов, тем самым освобождая неиспользованную память
CopyTo	Копирует все элементы множества в указанный массив, начиная с определенного индекса
Clear	Удаляет все элементы множества
Remove	Удаляет указанный элемент из множества
ExceptWith	Выполняет операцию разности текущей и переданной коллекции (массива)

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
IntersectWith	Выполняет операцию пересечения текущей и переданной коллекции (массива)
SymmetricExceptWith	Выполняет операцию симметричной разности текущей и переданной коллекции (массива)
UnionWith	Выполняет операцию объединения текущей и переданной коллекции (массива)
IsProperSubsetOf	Определяет, является ли текущее множество строгим подмножеством заданной коллекции или массива
IsProperSupersetOf	Определяет, является ли текущее множество строгим надмножеством заданной коллекции или массива
IsSubsetOf	Определяет, является ли текущее множество подмножеством заданной коллекции или массива
IsSupersetOf	Определяет, является ли текущее множество надмножеством заданной коллекции или массива
Overlaps	Определяет, пересекается ли текущее множество с заданной коллекцией или массивом
SetEquals	Определяет, содержит ли текущее множество все элементы из заданной коллекции или массива

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в множество.

```
bool Add(
   T  value     // значение элемента
   );
```

Параметры

value

[in] Значение элемента для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в множестве.

```
int Count();
```

Возвращаемое значение

Возвращает количество элементов.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли множество элемент с указанным значением.

```
bool Contains(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает true, если во множестве есть элемент с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

Comparer

Возвращает указатель на интерфейс IEqualityComparer<T>, использующийся для организации множества.

```
IEqualityComparer<T>* Comparer() const;
```

Возвращаемое значение

Возвращает указатель на интерфейс IEqualityComparer<T>.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

Сокращает емкость множества до фактического числа элементов, тем самым освобождая неиспользованную память.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все элементы множества в указанный массив, начиная с определенного индекса.

```
int CopyTo(
   T&          dst_array[],      // массив для записи
   const int   dst_start=0       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны элементы множества.

dst_start=0

[in] Индекс в массиве, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных элементов.

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы множества.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет указанный элемент из множества.

```
bool Remove(
   T  item     // значение элемента
   );
```

Параметры

item

[in] Значение элемента, которое нужно удалить.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

ExceptWith

Выполняет операцию разности текущей и переданной коллекции (массива). То есть удаляет из текущей коллекции (массива) все элементы, которые есть также и в указанной коллекции (массиве).

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void ExceptWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, которая будет вычитаться из текущего множества.

&collection[]

[in] Массив, который будет вычитаться из текущего множества.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

IntersectWith

Выполняет операцию пересечения текущей и переданной коллекции (массива). То есть оставляет в текущей коллекции только те элементы, которые имеются также и в указанной коллекции (массиве).

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void IntersectWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, с которой будет строиться произведение.

&collection[]

[in] Массив, с которым будет строиться произведение.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

Выполняет операцию симметричной разности текущей и переданной коллекции (массива). То есть в текущей коллекции (массиве) будут содержатся только те элементы, которые имелись либо в исходном объекте, либо в переданном, но не одновременно в них обоих.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void SymmetricExceptWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, с которой будет строиться симметричная разность.

&collection[]

[in] Массив, с которым будет строиться симметричная разность.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

UnionWith

Выполняет операцию объединения текущей и переданной коллекции (массива). То есть добавляет в текущую коллекцию (массив) отсутствующие элементы из указанной коллекции (массива).

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void UnionWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, с которой будет строиться сумма.

&collection[]

[in] Массив, с которым будет строиться сумма.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

Определяет, является ли текущее множество строгим подмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsProperSubsetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true в случае, если текущее множество является строгим подмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

Определяет, является ли текущее множество строгим надмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsProperSupersetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true в случае, если текущее множество является строгим надмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsSubsetOf

Определяет, является ли текущее множество подмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsSubsetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true в случае, если текущее множество является подмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

Определяет, является ли текущее множество надмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsSupersetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true в случае, если текущее множество является надмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

Overlaps

Определяет, пересекается ли текущее множество с заданной коллекцией или массивом.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // коллекция для сравнения
   );
```

Версия для работы с массивом.

```
bool Overlaps(
   T&  array[]                     // массив для сравнения
   );
```

Параметры

**collection*

[in] Коллекция для определения пересечения.

&collection[]

[in] Массив для определения пересечения.

Возвращаемое значение

Возвращает true в случае, если между текущим множеством и коллекцией или массивом есть пересечение, иначе false.

© 2000-2025, MetaQuotes Ltd.

SetEquals

Определяет, содержит ли текущее множество все элементы из заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // коллекция для сравнения
   );
```

Версия для работы с массивом.

```
bool SetEquals(
   T&  array[]                     // массив для сравнения
   );
```

Параметры

**collection*

[in] Коллекция для сопоставления элементов.

&collection[]

[in] Массив для сопоставления элементов.

Возвращаемое значение

Возвращает true в случае, если в текущем множестве есть все элементы указанной коллекции или массива, иначе false.

© 2000-2025, MetaQuotes Ltd.

CLinkedList<T>

Класс CLinkedList<T> — шаблонный класс, реализующий интерфейс ICollection<T>.

Описание

Класс CLinkedList<T> является реализацией динамического двунаправленного списка данных типа T. Данный класс предоставляет основные методы для работы с двунаправленными списками: добавление, удаление, поиск элемента и другие.

Декларация

```
template<typename T>
class CLinkedList : public ICollection<T>
```

Заголовок

```
#include <Generic\LinkedList.mqh>
```

Иерархия наследования

ICollection

CLinkedList

Методы класса

Метод	Описание
Add	Добавляет элемент в двунаправленный список
AddAfter	Добавляет элемент в двунаправленный список после указанного узла
AddBefore	Добавляет элемент в двунаправленный список до указанного узла
AddFirst	Добавляет элемент в начало двунаправленного списка
AddLast	Добавляет элемент в конец двунаправленного списка
Count	Возвращает количество элементов в двунаправленном списке
Head	Возвращает указатель на первый узел двунаправленного списка
First	Возвращает указатель на первый узел двунаправленного списка
Last	Возвращает указатель на последний узел двунаправленного списка
Contains	Определяет, содержит ли двунаправленный список элемент с указанным значением
CopyTo	Копирует все элементы двунаправленного списка в указанный массив, начиная с определенного индекса
Clear	Удаляет все элементы коллекции
Remove	Удаляет первое вхождение указанного элемента из двунаправленного списка

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
RemoveFirst	Удаляет первый элемент из двунаправленного списка
RemoveLast	Удаляет последний элемент из двунаправленного списка
Find	Ищет первое вхождение заданного значения в двунаправленный список
FindLast	Ищет последнее вхождение заданного значения в двунаправленный список

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в двунаправленный список.

```
bool Add(
   T  value     // значение элемента
   );
```

Параметры

value

[in] Значение элемента для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

AddAfter

Добавляет элемент в двунаправленный список после указанного узла.

Версия для добавления элемента по значению.

```
CLinkedListNode<T>* AddAfter(
   CLinkedListNode<T>*   node,         // узел, после которого следует добавление
   T                     value         // добавляемый элемент
   );
```

Возвращаемое значение

Возвращает указатель на добавленный узел.

Версия для добавления элемента как сформированного узла по значению.

```
bool AddAfter(
   CLinkedListNode<T>*   node,         // узел, после которого следует добавление
   CLinkedListNode<T>*   new_node      // добавляемый узел
   );
```

Параметры

**node*

[in] Узел двунаправленного списка, после которого будет добавлен новый элемент.

value

[in] Элемент для добавления.

**new_node*

[in] Узел для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

AddBefore

Добавляет элемент в двунаправленный список до указанного узла.

Версия для добавления элемента по значению.

```
CLinkedListNode<T>* AddBefore(
   CLinkedListNode<T>*   node,         // узел, до которого следует добавление
   T                     value         // добавляемый элемент
   );
```

Возвращаемое значение

Возвращает указатель на добавленный узел.

Версия для добавления элемента как сформированного узла по значению.

```
bool AddBefore(
   CLinkedListNode<T>*   node,         // узел, до которого следует добавление
   CLinkedListNode<T>*   new_node      // добавляемый узел
   );
```

Параметры

node

[in] Узел двунаправленного списка, до которого будет добавлен новый элемент.

value

[in] Элемент для добавления.

new_node

[in] Узел для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

AddFirst

Добавляет элемент в начало двунаправленного списка.

Версия для добавления элемента по значению.

```
CLinkedListNode<T>* AddFirst(
   T    value                     // элемент для добавления
   );
```

Возвращаемое значение

Возвращает указатель на добавленный узел.

Версия для добавления элемента как сформированного узла по значению.

```
bool AddFirst(
   CLinkedListNode<T>*  node      // узел для добавления
   );
```

Параметры

value

[in] Элемент для добавления.

**node*

[in] Узел для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

AddLast

Добавляет элемент в конец двунаправленного списка.

Версия для добавления элемента по значению.

```
CLinkedListNode<T>* AddLast(
   T  value                        // элемент для добавления
   );
```

Возвращаемое значение

Возвращает указатель на добавленный узел.

Версия для добавления элемента, как сформированного узла по значению.

```
bool AddLast(
   CLinkedListNode<T>*  node      // узел для добавления
   );
```

Параметры

value

[in] Элемент для добавления.

**node*

[in] Узел для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в двунаправленном списке.

```
int Count();
```

Возвращаемое значение

Возвращает количество элементов.

© 2000-2025, MetaQuotes Ltd.

Head

Возвращает указатель на первый узел двунаправленного списка.

```
CLinkedListNode<T>* Head();
```

Возвращаемое значение

Возвращает указатель на первый узел.

© 2000-2025, MetaQuotes Ltd.

First

Возвращает указатель на первый узел двунаправленного списка.

```
CLinkedListNode<T>* First();
```

Возвращаемое значение

Возвращает указатель на первый узел.

© 2000-2025, MetaQuotes Ltd.

Last

Возвращает указатель на последний узел двунаправленного списка.

```
CLinkedListNode<T>* Last();
```

Возвращаемое значение

Возвращает указатель на последний узел.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли двунаправленный список элемент с указанным значением.

```
bool Contains(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает true, если в двунаправленном списке есть элемент с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все элементы двунаправленного списка в указанный массив, начиная с определенного индекса.

```
int CopyTo(
   T&          dst_array[],      // массив для записи
   const int   dst_start=0       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны элементы двунаправленного списка.

dst_start=0

[in] Индекс в массиве, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных элементов.

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы коллекции.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет первое вхождение указанного элемента из двунаправленного списка.

Версия для удаления элемента по значению.

```
bool Remove(
   T  item                          // значение элемента
   );
```

Версия для удаления элемента по указателю на узел.

```
bool Remove(
   CLinkedListNode<T>*  node      // узел элемента
   );
```

Параметры

item

[in] Значение элемента, которое нужно удалить.

**node*

[in] Узел элемента, который нужно удалить.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

RemoveFirst

Удаляет первый элемент из двунаправленного списка.

```
bool RemoveFirst();
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

RemoveLast

Удаляет последний элемент из двунаправленного списка.

```
bool RemoveLast();
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Find

Ищет первое вхождение заданного значения в двунаправленный список.

```
CLinkedListNode<T>* Find(
   T  value     // искомое значение
   );
```

Параметры

value

[in] Искомое значение.

Возвращаемое значение

Возвращает указатель на первый найденный узел, содержащий искомое значение в случае успеха, иначе NULL.

© 2000-2025, MetaQuotes Ltd.

FindLast

Ищет последнее вхождение заданного значения в двунаправленный список.

```
CLinkedListNode<T>* FindLast(
   T  value     // искомое значение
   );
```

Параметры

value

[in] Искомое значение.

Возвращаемое значение

Возвращает указатель на последний найденный узел, содержащий искомое значение, в случае успеха, иначе NULL.

© 2000-2025, MetaQuotes Ltd.

CQueue<T>

Класс CQueue<T> — шаблонный класс, реализующий интерфейс ICollection<T>.

Описание

Класс CQueue<T> — динамическая коллекция данных типа T, организованная в виде очереди, который работает по принципу "первым вошёл — первым вышел" (FIFO).

Декларация

```
template<typename T>
class CQueue : public ICollection<T>
```

Заголовок

```
#include <Generic\Queue.mqh>
```

Иерархия наследования

ICollection

CQueue

Методы класса

Метод	Описание
Add	Добавляет элемент в очередь
Enqueue	Добавляет элемент в очередь
Count	Возвращает количество элементов в очереди
Contains	Определяет, содержит ли очередь элемент с указанным значением
TrimExcess	Сокращает емкость очереди до фактического числа элементов, тем самым освобождая неиспользованную память
CopyTo	Копирует все элементы очереди в указанный массив, начиная с определенного индекса
Clear	Удаляет все элементы очереди
Remove	Удаляет первое вхождение указанного элемента из очереди
Dequeue	Возвращает начальный элемент, удаляя его из очереди
Peek	Возвращает начальный элемент, не удаляя его из очереди

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в очередь.

```
bool Add(
   T  value     // значение элемента
   );
```

Параметры

value

[in] Значение элемента для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Enqueue

Добавляет элемент в очередь.

```
bool Enqueue(
   T  value     // элемент для добавления
   );
```

Параметры

value

[in] Элемент для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в очереди.

```
int Count();
```

Возвращаемое значение

Возвращает количество элементов.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли очередь элемент с указанным значением.

```
bool Contains(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает true, если в очереди есть элемент с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

Сокращает емкость очереди до фактического числа элементов, тем самым освобождая неиспользованную память.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все элементы очереди в указанный массив, начиная с определенного индекса.

```
int CopyTo(
   T&          dst_array[],      // массив для записи
   const int   dst_start=0       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны элементы очереди.

dst_start=0

[in] Индекс в массиве, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных элементов.

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы очереди.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет первое вхождение указанного элемента из очереди.

```
bool Remove(
   T  item     // значение элемента
   );
```

Параметры

item

[in] Значение элемента, которое нужно удалить.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Dequeue

Возвращает начальный элемент, удаляя его из очереди.

```
T Dequeue();
```

Возвращаемое значение

Возвращает начальный элемент.

© 2000-2025, MetaQuotes Ltd.

Peek

Возвращает начальный элемент, не удаляя его из очереди.

```
T Peek();
```

Возвращаемое значение

Возвращает начальный элемент.

© 2000-2025, MetaQuotes Ltd.

CRedBlackTree<T>

Класс CRedBlackTree<T> — шаблонный класс, реализующий интерфейс ICollection<T>.

Описание

Класс CRedBlackTree<T> — реализация динамического красно-черного дерева, в узлах которого хранятся данные типа T. Класс предоставляет основные методы для работы с красно-черными деревьями: добавление, удаление, поиск максимального, минимального значения и другие.

Декларация

```
template<typename T>
class CRedBlackTree : public ICollection<T>
```

Заголовок

```
#include <Generic\RedBlackTree.mqh>
```

Иерархия наследования

ICollection

CRedBlackTree

Методы класса

Метод	Описание
Add	Добавляет элемент в красно-черное дерево
Root	Возвращает указатель на корень красно-черного дерева
Count	Возвращает количество элементов в красно-черном дереве
Contains	Определяет, содержит ли красно-черное дерево элемент с указанным значением
Comparer	Возвращает указатель на интерфейс IComparer<T>, использующийся для организации красно-черного дерева
TryGetMin	Получает минимальный элемент из красно-черного дерева
TryGetMax	Получает максимальный элемент из красно-черного дерева
CopyTo	Копирует все элементы красно-черного дерева в указанный массив, начиная с определенного индекса
Clear	Удаляет все элементы красно-черного дерева
Remove	Удаляет вхождение указанного элемента из красно-черного дерева
RemoveMin	Удаляет элемент с минимальным значением из красно-черного дерева

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
RemoveMax	Удаляет элемент с максимальным значением из красно-черного дерева
Find	Ищет вхождение заданного значения в красно-черное дерево
FindMax	Ищет элемент с максимальным значением в красно-черное дерево
FindMin	Ищет элемент с минимальным значением в красно-черном дереве

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в красно-черное дерево.

```
bool Add(
   T  value     // элемент для добавления
   );
```

Параметры

value

[in] Элемент для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в красно-черном дереве.

```
int Count();
```

Возвращаемое значение

Возвращает количество элементов.

© 2000-2025, MetaQuotes Ltd.

Root

Возвращает указатель на корень красно-черного дерева.

```
CRedBlackTreeNode<T>* Root();
```

Возвращаемое значение

Возвращает указатель на корень.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли красно-черное дерево элемент с указанным значением.

```
bool Contains(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает true если в красно-черном дереве есть элемент с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

Comparer

Возвращает указатель на интерфейс IComparer<T>, использующийся для организации красно-черного дерева.

```
IComparer<T>* Comparer() const;
```

Возвращаемое значение

Возвращает указатель на интерфейс IComparer<T>.

© 2000-2025, MetaQuotes Ltd.

TryGetMin

Получает минимальный элемент из красно-черного дерева.

```
bool TryGetMin(
   T&  min     // переменная для записи значения
   );
```

Параметры

&min

[out] Переменная, в которую будет записано минимальное значение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TryGetMax

Получает максимальный элемент из красно-черного дерева.

```
bool TryGetMax(
   T&  max      // переменная для записи значения
   );
```

Параметры

&max

[out] Переменная, в которую будет записано максимальное значение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все элементы красно-черного дерева в указанный массив, начиная с определенного индекса.

```
int CopyTo(
   T&          dst_array[],      // массив для записи
   const int   dst_start=0       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны элементы красно-черного дерева.

dst_start=0

[in] Индекс в массиве, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных элементов.

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы красно-черного дерева.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет вхождение указанного элемента из красно-черного дерева.

Версия для удаления элемента по заданному значению.

```
bool Remove(
   T  value                          // значение элемента
   );
```

Версия для удаления элемента по указателю на узел.

```
bool Remove(
   CRedBlackTreeNode<T>*  node      // узел элемента
   );
```

Параметры

item

[in] Значение элемента, которое нужно удалить.

**node*

[in] Узел элемента, который нужно удалить.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

RemoveMin

Удаляет элемент с минимальным значением из красно-черного дерева.

```
bool RemoveMin();
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

RemoveMax

Удаляет элемент с максимальным значением из красно-черного дерева.

```
bool RemoveMax();
```

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Find

Ищет вхождение заданного значения в красно-черное дерево.

```
CRedBlackTreeNode<T>* Find(
   T  value     // искомое значение
   );
```

Параметры

value

[in] Искомое значение.

Возвращаемое значение

Возвращает указатель на найденный узел, содержащий искомое значение в случае успеха, иначе NULL.

© 2000-2025, MetaQuotes Ltd.

FindMin

Ищет элемент с минимальным значением в красно-черном дереве.

```
CRedBlackTreeNode<T>* FindMin();
```

Возвращаемое значение

Возвращает указатель на узел, содержащий минимальное значение в случае успеха, иначе NULL.

© 2000-2025, MetaQuotes Ltd.

FindMax

Ищет элемент с максимальным значением в красно-черное дерево.

```
CRedBlackTreeNode<T>* FindMax();
```

Возвращаемое значение

Возвращает указатель на узел, содержащий максимальное значение в случае успеха, иначе NULL.

© 2000-2025, MetaQuotes Ltd.

CSortedMap<TKey,TValue>

Класс CSortedMap<TKey,TValue> — шаблонный класс, реализующий интерфейс IMap<TKey,TValue>.

Описание

Класс CSortedMap<TKey,TValue> — реализация динамической хэш-таблицы, данные которой хранятся в виде пар "ключ — значение", отсортированных по ключу и с соблюдением требования уникальности ключа. Данный класс предоставляет основные методы для работы с хэш-таблицей: доступ к значению по ключу, поиск и удаление пары "ключ-значение" и другие.

Декларация

```
template<typename TKey, typename TValue>
class CSortedMap : public IMap<TKey, TValue>
```

Заголовок

```
#include <Generic\SortedMap.mqh>
```

Иерархия наследования

ICollection
  IMap
    CSortedMap

Методы класса

Метод	Описание
Add	Добавляет пару "ключ — значение" в хэш-таблицу
Count	Возвращает количество элементов в отсортированной хэш-таблице
Contains	Определяет, содержит ли сортированная хэш-таблица указанную пару "ключ — значение"
ContainsKey	Определяет, содержит ли сортированная хэш-таблица пару "ключ — значение" с указанным ключом
ContainsValue	Определяет, содержит ли сортированная хэш-таблица пару "ключ — значение" с указанным значением
Comparer	Возвращает указатель на интерфейс IComparer<T>, использующийся для организации сортированной хэш-таблицы
CopyTo	Копирует все пары "ключ — значение" из сортированной хэш-таблицы в указанные массивы, начиная с определенного индекса
Clear	Удаляет все элементы сортированной хэш-таблицы
Remove	Удаляет первое вхождение пары "ключ — значение" сортированной хэш-таблицы

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
TryGetValue	Получает элемент из сортированной хэш-таблицы по заданному ключу
TrySetValue	Изменяет значение пары "ключ — значение" из отсортированной хэш-таблицы по заданному ключу

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет пару "ключ — значение" в хэш-таблицу.

Версия для добавления сформированной пары "ключ — значение".

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // пара "ключ — значение"
   );
```

Версия для добавления новой пары "ключ — значение" с указанным ключом и значением.

```
bool Add(
   TKey    key,                         // ключ
   TValue  value                        // значение
   );
```

Параметры

**pair*

[in] Пара "ключ — значение".

key

[in] Ключ.

value

[in] Значение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в отсортированной хэш-таблице.

```
int Count();
```

© 2000-2025, MetaQuotes Ltd.

Comparer

Возвращает указатель на интерфейс IComparer<T>, использующийся для организации сортированной хэш-таблицы.

```
IComparer<TKey>* Comparer() const;
```

Возвращаемое значение

Возвращает указатель на интерфейс IComparer<T>.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли сортированная хэш-таблица указанную пару "ключ — значение".

Версия для работы с сформированной парой "ключ — значение".

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // пара "ключ — значение"
   );
```

Версия для работы с парой "ключ — значение" в виде отдельно заданного ключа и значения.

```
bool Contains(
   TKey    key,                         // ключ
   TValue  value                        // значение
   );
```

Параметры

**item*

[in] Пара "ключ — значение".

key

[in] Ключ.

value

[in] Значение.

Возвращаемое значение

Возвращает true, если в сортированной хэш-таблице есть пара "ключ — значение" с указанным ключом и значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

ContainsKey

Определяет, содержит ли сортированная хэш-таблица пару "ключ — значение" с указанным ключом.

```
bool ContainsKey(
   TKey  key      // ключ
   );
```

Параметры

key

[in] Ключ.

Возвращаемое значение

Возвращает true, если в сортированной хэш-таблице есть пара "ключ — значение" с указанным ключом, иначе false.

© 2000-2025, MetaQuotes Ltd.

ContainsValue

Определяет, содержит ли сортированная хэш-таблица пару "ключ — значение" с указанным значением.

```
bool ContainsValue(
   TValue  value     // значение
   );
```

Параметры

value

[in] Значение.

Возвращаемое значение

Возвращает true, если в сортированной хэш-таблице есть пара "ключ — значение" с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все пары "ключ — значение" из сортированной хэш-таблицы в указанные массивы, начиная с определенного индекса.

Версия для копирования сортированной хэш-таблицы в массив пар "ключ-значение".

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // массив для записи пар "ключ-значен
   const int                    dst_start=0      // начальный индекс для записи
   );
```

Версия для копирования сортированной хэш-таблицы в отдельные массивы для ключей и значений.

```
int CopyTo(
   TKey&       dst_keys[],                       // массив для записи ключей
   TValue&     dst_values[],                     // массив для записи значений
   const int   dst_start=0                       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны все пары из хэш-таблицы.

&dst_keys[]

[out] Массив, в который будут записаны все ключи из хэш-таблицы.

&dst_values[]

[out] Массив, в который будут записаны все значения из хэш-таблицы.

dst_start=0

[in] Индекс в массивах, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных пар "ключ-значение".

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы сортированной хэш-таблицы.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет первое вхождение пары "ключ — значение" сортированной хэш-таблицы.

Версия для удаления пары "ключ — значение" по сформированной паре "ключ — значение".

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // пара "ключ — значение"
   );
```

Версия для удаления пары "ключ — значение" по ключу.

```
bool Remove(
   TKey  key                           // ключ
   );
```

Параметры

**item*

[in] Пара "ключ — значение".

key

[in] Ключ.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TryGetValue

Получает элемент из сортированной хэш-таблицы по заданному ключу.

```
bool TryGetValue(
   TKey      key,       // ключ
   TValue&   value      // переменная для записи значения
   );
```

Параметры

key

[in] Ключ.

&value

[out] Переменная, в которую будет записано указанное значение пары "ключ — значение".

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

Изменяет значение пары "ключ — значение" из сортированной хэш-таблицы по заданному ключу.

```
bool TrySetValue(
   TKey    key,       // ключ
   TValue  value      // новое значение
   );
```

Параметры

key

[in] Ключ.

value

[in] Новое значение, которое нужно присвоить указанной паре "ключ — значение".

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

CSortedSet<T>

Класс CSortedSet<T> — шаблонный класс, реализующий интерфейс ISet<T>.

Описание

Класс CSortedSet<T> является реализацией отсортированного динамического множества данных типа T, с соблюдением требования уникальности каждого значения. Данный класс предоставляет основные методы для работы с множествами и операциями над ними: объединение и пересечение множеств, определение строгих и нестрогих подмножеств и другие.

Декларация

```
template<typename T>
class CSortedSet : public ISet<T>
```

Заголовок

```
#include <Generic\SortedSet.mqh>
```

Иерархия наследования

ICollection
  ISet
    CSortedSet

Методы класса

Метод	Описание
Add	Добавляет элемент в сортированное множество
Count	Возвращает количество элементов в сортированном множестве
Contains	Определяет, содержит ли сортированное множество элемент с указанным значением
Comparer	Возвращает указатель на интерфейс IComparer<T>, использующийся для организации сортированного множества
TryGetMin	Получает минимальный элемент из сортированного множества
TryGetMax	Получает максимальный элемент из сортированного множества
CopyTo	Копирует все элементы сортированного множества в указанный массив, начиная с определенного индекса
Clear	Удаляет все элементы сортированного множества
Remove	Удаляет вхождение указанного элемента из сортированного множества
ExceptWith	Выполняет операцию разности текущей и переданной коллекции (массива)

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
IntersectWith	Выполняет операцию пересечения текущей и переданной коллекции (массива)
SymmetricExceptWith	Выполняет операцию симметричной разности текущей и переданной коллекции (массива)
UnionWith	Выполняет операцию объединения текущей и переданной коллекции (массива)
IsProperSubsetOf	Определяет, является ли текущее сортированное множество строгим подмножеством заданной коллекции или массива
IsProperSupersetOf	Определяет, является ли текущее сортированное множество строгим надмножеством заданной коллекции или массива
IsSubsetOf	Определяет, является ли текущее сортированное множество подмножеством заданной коллекции или массива
IsSupersetOf	Определяет, является ли текущее сортированное множество надмножеством заданной коллекции или массива
Overlaps	Определяет, пересекается ли текущее сортированное множество с заданной коллекцией или массивом
SetEquals	Определяет, содержит ли текущее сортированное множество все элементы из заданной коллекции или массива
GetViewBetween	Получает от текущего сортированного множества подмножество, заданное минимальным и максимальным значением
GetReverse	Получает копию текущего сортированного множества, где все элементы расположены в обратном порядке

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в сортированное множество.

```
bool Add(
   T  value     // значение элемента
   );
```

Параметры

value

[in] Значение элемента для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в сортированном множестве.

```
int Count();
```

Возвращаемое значение

Возвращает количество элементов.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли сортированное множество элемент с указанным значением.

```
bool Contains(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает true, если во множестве есть элемент с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

Comparer

Возвращает указатель на интерфейс IComparer<T>, использующийся для организации сортированного множества.

```
IComparer<T>* Comparer() const;
```

Возвращаемое значение

Возвращает указатель на интерфейс IComparer<T>.

© 2000-2025, MetaQuotes Ltd.

TryGetMin

Получает минимальный элемент из сортированного множества.

```
bool TryGetMin(
   T&  min     // переменная для записи значения
   );
```

Параметры

&min

[out] Переменная, в которую будет записано минимальное значение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

TryGetMax

Получает максимальный элемент из сортированного множества.

```
bool TryGetMax(
   T&  max      // переменная для записи значения
   );
```

Параметры

&max

[out] Переменная, в которую будет записано максимальное значение.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все элементы сортированного множества в указанный массив, начиная с определенного индекса.

```
int CopyTo(
   T&          dst_array[],      // массив для записи
   const int   dst_start=0       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны элементы множества.

dst_start=0

[in] Индекс в массиве, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных элементов.

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы сортированного множества.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет вхождение указанного элемента из сортированного множества.

```
bool Remove(
   T  item     // значение элемента
   );
```

Параметры

item

[in] Значение элемента, которое нужно удалить.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

ExceptWith

Выполняет операцию разности текущей и переданной коллекции (массива). То есть удаляет из текущей коллекции (массива) все элементы, которые есть также и в указанной коллекции (массиве).

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void ExceptWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, которая будет вычитаться из текущего сортированного множества.

&collection[]

[in] Массив, который будет вычитаться из текущего сортированного множества.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

IntersectWith

Выполняет операцию пересечения текущей и переданной коллекции (массива). То есть оставляет в текущей коллекции только те элементы, которые имеются также и в указанной коллекции (массиве).

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void IntersectWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, с которой будет строиться произведение.

&collection[]

[in] Массив, с которым будет строиться произведение.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

Выполняет операцию симметричной разности текущей и переданной коллекции (массива). То есть в текущей коллекции (массиве) будут содержатся только те элементы, которые имелись либо в исходном объекте, либо в переданном, но не одновременно в них обоих.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void SymmetricExceptWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, с которой будет строиться симметричная разность.

&collection[]

[in] Массив, с которым будет строиться симметричная разность.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

UnionWith

Выполняет операцию объединения текущей и переданной коллекции (массива). То есть добавляет в текущую коллекцию (массив) отсутствующие элементы из указанной коллекции (массива).

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // коллекция
   );
```

Версия для работы с массивом.

```
void UnionWith(
   T&  array[]                     // массив
   );
```

Параметры

**collection*

[in] Коллекция, с которой будет строиться сумма.

&collection[]

[in] Массив, с которым будет строиться сумма.

Примечание

Результат записывается в текущую коллекцию (массив).

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

Определяет, является ли текущее сортированное множество строгим подмножеством заданной коллекции или массива.

Версия для работы с коллекцией реализующей интерфейс ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsProperSubsetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true, если текущее сортированное множество является строгим подмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

Определяет, является ли текущее сортированное множество строгим надмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsProperSupersetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true в случае, если текущее сортированное множество является строгим надмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsSubsetOf

Определяет, является ли текущее сортированное множество подмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsSubsetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true в случае, если текущее сортированное множество является подмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

Определяет, является ли текущее сортированное множество надмножеством заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // коллекция для определения отношения
   );
```

Версия для работы с массивом.

```
bool IsSupersetOf(
   T&  array[]                     // массив для определения отношения
   );
```

Параметры

**collection*

[in] Коллекция для определения отношения.

&collection[]

[in] Массив для определения отношения.

Возвращаемое значение

Возвращает true в случае, если текущее сортированное множество является надмножеством, иначе false.

© 2000-2025, MetaQuotes Ltd.

Overlaps

Определяет, пересекается ли текущее сортированное множество с заданной коллекцией или массивом.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // коллекция для сравнения
   );
```

Версия для работы с массивом.

```
bool Overlaps(
   T&  array[]                     // массив для сравнения
   );
```

Параметры

**collection*

[in] Коллекция для определения пересечения.

&collection[]

[in] Массив для определения пересечения.

Возвращаемое значение

Возвращает true в случае, если между текущим сортированным множеством и коллекцией или массивом есть пересечение, иначе false.

© 2000-2025, MetaQuotes Ltd.

SetEquals

Определяет, содержит ли текущее сортированное множество все элементы из заданной коллекции или массива.

Версия для работы с коллекцией, реализующей интерфейс ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // коллекция для сравнения
   );
```

Версия для работы с массивом.

```
bool SetEquals(
   T&  array[]                     // массив для сравнения
   );
```

Параметры

**collection*

[in] Коллекция для сопоставления элементов.

&collection[]

[in] Массив для сопоставления элементов.

Возвращаемое значение

Возвращает true в случае, если в текущем сортированном множестве есть все элементы указанной коллекции или массива, иначе false.

© 2000-2025, MetaQuotes Ltd.

GetViewBetween

Получает от текущего сортированного множества подмножество, заданное минимальным и максимальным значением.

```
bool GetViewBetween(
   T&  array[],          // массив для записи
   T   lower_value,      // минимальное значение
   T   upper_value       // максимальное значение
   );
```

Параметры

&array[]

[out] Массив для записи подмножества.

lower_value

[in] Минимальное значение диапазона.

upper_value

[in] Максимальное значение диапазона.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

GetReverse

Получает копию текущего сортированного множества, где все элементы расположены в обратном порядке.

```
bool GetReverse(
   T&  array[]     // массив для записи
   );
```

Параметры

&array[]

[out] Массив для записи.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

CStack<T>

Класс CStack<T> — шаблонный класс, реализующий интерфейс ICollection<T>.

Описание

Класс CStack<T> — динамическая коллекция данных типа T, организованная в виде стека, который работает по принципу "последним пришел — первым вышел" (LIFO).

Декларация

```
template<typename T>
class CStack : public ICollection<T>
```

Заголовок

```
#include <Generic\Stack.mqh>
```

Иерархия наследования

ICollection

CStack

Методы класса

Метод	Описание
Add	Добавляет элемент в стек
Count	Возвращает количество элементов в стеке
Contains	Определяет, содержит ли стек элемент с указанным значением
TrimExcess	Сокращает емкость стека до фактического числа элементов
CopyTo	Копирует все элементы стека в указанный массив, начиная с определенного индекса
Clear	Удаляет все элементы стека
Remove	Удаляет первое вхождение указанного элемента из стека
Push	Добавляет элемент в стек
Peek	Возвращает головной элемент, не удаляя его из стека
Pop	Возвращает головной элемент, удаляя его из стека

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в стек.

```
bool Add(
   T  value     // значение элемента
   );
```

Параметры

value

[in] Значение элемента для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Count

Возвращает количество элементов в стеке.

```
int Count();
```

Возвращаемое значение

Возвращает количество элементов.

© 2000-2025, MetaQuotes Ltd.

Contains

Определяет, содержит ли стек элемент с указанным значением.

```
bool Contains(
   T  item     // искомое значение
   );
```

Параметры

item

[in] Искомое значение.

Возвращаемое значение

Возвращает true, если в стеке есть элемент с указанным значением, иначе false.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

Сокращает емкость стека до фактического числа элементов, тем самым освобождая неиспользуемую память.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

Копирует все элементы стека в указанный массив, начиная с определенного индекса.

```
int CopyTo(
   T&          dst_array[],      // массив для записи
   const int   dst_start=0       // начальный индекс для записи
   );
```

Параметры

&dst_array[]

[out] Массив, в который будут записаны элементы стека.

dst_start=0

[in] Индекс в массиве, с которого начинается копирование.

Возвращаемое значение

Возвращает количество скопированных элементов.

© 2000-2025, MetaQuotes Ltd.

Clear

Удаляет все элементы стека.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет первое вхождение указанного элемента из стека.

```
bool Remove(
   T  item     // значение элемента
   );
```

Параметры

item

[in] Значение элемента, которое нужно удалить.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Push

Добавляет элемент в стек.

```
bool Push(
   T  value     // элемент для добавления
   );
```

Параметры

value

[in] Элемент для добавления.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Peek

Возвращает головной элемент, не удаляя его из стека.

```
T Peek();
```

Возвращаемое значение

Возвращает головной элемент.

© 2000-2025, MetaQuotes Ltd.

Pop

Возвращает головной элемент, удаляя его из стека.

```
T Pop();
```

Возвращаемое значение

Возвращает головной элемент.

© 2000-2025, MetaQuotes Ltd.

ArrayBinarySearch

Ищет указанное значение в отсортированном по возрастанию одномерном массиве, используя интерфейс IComparable<T> для сравнения элементов.

```
template<typename T>
int ArrayBinarySearch(
   T&              array[],          // массив для поиска
   const int       start_index,      // начальный индекс
   const int       count,            // диапазон поиска
   T               value,            // искомое значение
   IComparer<T>*   comparer          // интерфейс для сравнения
   );
```

Параметры

&array[]

[out] Массив для поиска.

value

[in] Искомое значение.

**comparer*

[in] Интерфейс для сравнения элементов.

start_index

[in] Начальный индекс, с которого начинается поиск.

count

[in] Длина диапазона поиска.

Возвращаемое значение

Возвращает индекс найденного элемента. Если искомое значение не найдено, то возвращает индекс меньшего элемента, которое ближе всех по значению.

© 2000-2025, MetaQuotes Ltd.

ArrayIndexOf

Ищет первое вхождение значения в одномерном массиве.

```
template<typename T>
int ArrayIndexOf(
   T&          array[],          // массив для поиска
   T           value,            // искомое значение
   const int   start_index,      // начальный индекс
   const int   count             // диапазон поиска
   );
```

Параметры

&array[]

[out] Массив для поиска.

value

[in] Искомое значение.

start_index

[in] Начальный индекс, с которого начинается поиск.

count

[in] Длина диапазона поиска.

Возвращаемое значение

Возвращает индекс первого найденного элемента. Если значение не найдено, возвращает -1.

© 2000-2025, MetaQuotes Ltd.

ArrayLastIndexOf

Ищет последнее вхождение значения в одномерном массиве.

```
template<typename T>
int ArrayLastIndexOf(
   T&          array[],          // массив для поиска
   T           value,            // искомое значение
   const int   start_index,      // начальный индекс
   const int   count             // диапазон поиска
   );
```

Параметры

&array[]

[out] Массив для поиска.

value

[in] Искомое значение.

start_index

[in] Начальный индекс, с которого начинается поиск.

count

[in] Длина диапазона поиска.

Возвращаемое значение

Возвращает индекс последнего найденного элемента. Если значение не найдено, возвращает -1.

© 2000-2025, MetaQuotes Ltd.

ArrayReverse

Изменяет последовательность элементов в одномерном массиве.

```
template<typename T>
bool ArrayReverse(
   T&          array[],          // исходный массив
   const int   start_index,      // начальный индекс
   const int   count             // количество элементов
   );
```

Параметры

&array[]

[out] Исходный массив.

start_index

[in] Начальный индекс.

count

[in] Количество элементов массива, участвующих в операции.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

Compare

Сравнивает два значения на отношение "Больше, меньше или равно".

Версия для сравнения двух значений типа bool.

```
int Compare(
   const bool  x,          // первое значение
   const bool  y           // второе значение
   );
```

Версия для сравнения двух значений типа char.

```
int Compare(
   const char  x,          // первое значение
   const char  y           // второе значение
   );
```

Версия для сравнения двух значений типа uchar.

```
int Compare(
   const uchar  x,         // первое значение
   const uchar  y          // второе значение
   );
```

Версия для сравнения двух значений типа short.

```
int Compare(
   const short  x,         // первое значение
   const short  y          // второе значение
   );
```

Версия для сравнения двух значений типа ushort.

```
int Compare(
   const ushort  x,        // первое значение
   const ushort  y         // второе значение
   );
```

Версия для сравнения двух значений типа color.

```
int Compare(
   const color  x,         // первое значение
   const color  y          // второе значение
   );
```

Версия для сравнения двух значений типа int.

```
int Compare(
   const int  x,           // первое значение
   const int  y            // второе значение
   );
```

© 2000-2025, MetaQuotes Ltd.

Версия для сравнения двух значений типа uint.

```
int Compare(
   const uint  x,          // первое значение
   const uint  y           // второе значение
   );
```

Версия для сравнения двух значений типа datetime.

```
int Compare(
   const datetime  x,      // первое значение
   const datetime  y       // второе значение
   );
```

Версия для сравнения двух значений типа long.

```
int Compare(
   const long  x,          // первое значение
   const long  y           // второе значение
   );
```

Версия для сравнения двух значений типа ulong.

```
int Compare(
   const ulong  x,         // первое значение
   const ulong  y          // второе значение
   );
```

Версия для сравнения двух значений типа float.

```
int Compare(
   const float  x,         // первое значение
   const float  y          // второе значение
   );
```

Версия для сравнения двух значений типа double.

```
int Compare(
   const double  x,        // первое значение
   const double  y         // второе значение
   );
```

Версия для сравнения двух значений типа string.

```
int Compare(
   const string  x,        // первое значение
   const string  y         // второе значение
   );
```

Версия для сравнения двух значений остальных типов.

```
template<typename T>
int Compare(
```

© 2000-2025, MetaQuotes Ltd.

```
   T  x,                  // первое значение
   T  y                   // второе значение
   );
```

Параметры

x

[in] Первое значение.

y

[in] Второе значение

Возвращаемое значение

Возвращает число, выражающее отношение двух сравниваемых значений:

- результат меньше нуля — x меньше y (x<y)
- результат равен нулю — x равен y (x=y)
- результат больше нуля — x больше y (x>y)

Примечание

Если тип T — объект, реализующий интерфейс IComparable<T>, то объекты будут сравниваться на основе его метода сравнения Compare. Во всех остальных случаях возвращается 0.

© 2000-2025, MetaQuotes Ltd.

Equals

Сравнивает два значения на равенство.

```
template<typename T>
bool Equals(
   T  x,      // первое значение
   T  y       // второе значение
   );
```

Параметры

x

[in] Первое значение.

y

[in] Второе значение.

Возвращаемое значение

Возвращает true, если объекты равны, иначе false.

Примечание

Если тип T — объект, реализующий интерфейс IEqualityComparable<T>, то объекты будут сравниваться на основе его метода сравнения Equals. Во всех остальных случаях применяется стандартное сравнение на равенство.

© 2000-2025, MetaQuotes Ltd.

GetHashCode

Вычисляет значение хэш-кода.

Версия для работы с типом bool.

```
int GetHashCode(
   const bool   value          // значение
   );
```

Версия для работы с типом char.

```
int GetHashCode(
   const char   value          // значение
   );
```

Версия для работы с типом uchar.

```
int GetHashCode(
   const uchar   value         // значение
   );
```

Версия для работы с типом short.

```
int GetHashCode(
   const short   value         // значение
   );
```

Версия для работы с типом ushort.

```
int GetHashCode(
   const ushort   value        // значение
   );
```

Версия для работы с типом color.

```
int GetHashCode(
   const color   value         // значение
   );
```

Версия для работы с типом int.

```
int GetHashCode(
   const int   value           // значение
   );
```

Версия для работы с типом uint.

```
int GetHashCode(
   const uint   value          // значение
   );
```

Версия для работы с типом datetime.

```
int GetHashCode(
```

© 2000-2025, MetaQuotes Ltd.

```
   const datetime   value      // значение
   );
```

Версия для работы с типом long.

```
int GetHashCode(
   const long   value          // значение
   );
```

Версия для работы с типом ulong.

```
int GetHashCode(
   const ulong   value         // значение
   );
```

Версия для работы с типом float.

```
int GetHashCode(
   const float   value         // значение
   );
```

Версия для работы с типом double.

```
int GetHashCode(
   const double   value        // значение
   );
```

Версия для работы с типом string.

```
int GetHashCode(
   const string   value        // значение
   );
```

Версия для работы с остальными типами.

```
template<typename T>
int GetHashCode(
   T   value                   // значение
   );
```

Параметры

value

[in] Значение, для которого нужно получить хэш-код.

Возвращаемое значение

Возвращает хэш-код.

Примечание

Если тип T — объект, реализующий интерфейс IEqualityComparable<T>, то хэш-код будет получен на основе его метода HashCode. Во всех остальных случаях хэш-код будет рассчитан как значение хэш-кода от имени типа данного значения.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Файловые операции

Этот раздел содержит технические детали работы с классами файловых операций и описание соответствующих компонентов стандартной библиотеки MQL5.

Использование классов файловых операций позволит сэкономить время при разработке приложений использующих файловый ввод/вывод.

Стандартная библиотека MQL5 (в части файловых операций) размещается в рабочем каталоге терминала в папке Include\Files.

Имя класса	Назначение класса
CFile	Базовый класс файловых операций
CFileBin	Класс операций с бинарным файлом
CFileTxt	Класс операций с текстовым файлом

© 2000-2025, MetaQuotes Ltd.

Класс CFile

Класс CFile является базовым классом для классов CFileBin и CFileTxt.

Описание

Класс CFile обеспечивает своим потомкам доступ к общим функциям API MQL5 по работе с файлами и папками.

Декларация

```
class CFile: public CObject
```

Заголовок

```
#include <Files\File.mqh>
```

Иерархия наследования

CObject

CFile

Прямые потомки

CFileBin, CFilePipe, CFileTxt

Методы класса по группам

Атрибуты	
Handle	Получает хэндл файла
Filename	Получает имя файла
Flags	Получает флаги файла
SetUnicode	Устанавливает/сбрасывает флаг FILE_UNICODE
SetCommon	Устанавливает/сбрасывает флаг FILE_COMMON
Общие методы работы с файлами	
Open	Открывает файл
Close	Закрывает файл
Delete	Удаляет файл
IsExist	Проверяет существование файла
Copy	Копирует файл
Move	Перемещает/переименовывает файл
Size	Получает размер файла

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
Tell	Получает текущее положение файлового указателя
Seek	Устанавливает положение файлового указателя
Flush	Сбрасывает на диск все данные
IsEnding	Определяет конец файла
IsLineEnding	Определяет конец строки
Общие методы работы с папками	
FolderCreate	Создает папку
FolderDelete	Удаляет папку
FolderClean	Очищает папку
Общие методы поиска файлов и папок	
FileFindFirst	Начинает поиск файлов
FileFindNext	Продолжает поиск файлов
FileFindClose	Закрывает хэндл поиска

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Handle

Получает хэндл открытого файла.

```
int  Handle()
```

Возвращаемое значение

Хэндл привязанного к экземпляру класса открытого файла. Если нет привязанного файла возвращается -1.

© 2000-2025, MetaQuotes Ltd.

FileName

Получает имя открытого файла.

```
string  FileName()
```

Возвращаемое значение

Имя привязанного к экземпляру класса открытого файла. Если нет привязанного файла возвращается "".

© 2000-2025, MetaQuotes Ltd.

Flags

Получает флаги открытия файла.

```
int  Flags()
```

Возвращаемое значение

Флаги открытия привязанного к экземпляру класса файла.

© 2000-2025, MetaQuotes Ltd.

SetUnicode

Устанавливает/сбрасывает флаг FILE_UNICODE.

```
void  SetUnicode(
   bool  unicode      // значение флага
   )
```

Параметры

unicode

[in] Новое значение флага FILE_UNICODE.

Примечание

От состояния флага FILE_UNICODE зависит работа со строками. Если флаг сброшен, то используется кодировка ANSI (однобайтовые символы). Если флаг установлен, то используется кодировка UNICODE (двухбайтовые символы). Если файл уже открыт, то флаг изменить нельзя.

© 2000-2025, MetaQuotes Ltd.

SetCommon

Устанавливает/сбрасывает флаг FILE_COMMON.

```
void  SetCommon(
   bool  common      // значение флага
   )
```

Параметры

common

[in] Новое значение флага FILE_COMMON.

Примечание

От состояния флага FILE_COMMON зависит, какая папка будет использоваться в качестве рабочей. Если флаг сброшен, то в качестве рабочей папки используется локальная папка терминала. Если флаг установлен, то в качестве рабочей папки используется общая папка. Если файл уже открыт, то флаг изменить нельзя.

© 2000-2025, MetaQuotes Ltd.

Open

Открывает указанный файл и, в случае удачного открытия, привязывает его к экземпляру класса.

```
int  Open(
   const string  file_name,        // имя файла
   int           flags,            // флаги
   short         delimiter=9       // разделитель
   )
```

Параметры

file_name

[in] Имя открываемого файла.

flags

[in] Флаги открытия файла.

delimiter=9

[in] Разделитель для CSV-файла.

Возвращаемое значение

Хэндл открытого файла.

Примечание

Рабочая папка определяется ранее установленным/сброшенным при помощи метода SetCommon() флагом.

© 2000-2025, MetaQuotes Ltd.

Close

Закрывает файл, привязанный к экземпляру класса.

```
void  Close()
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет файл, привязанный к экземпляру класса.

```
void  Delete()
```

Delete

Удаляет указанный файл.

```
void  Delete(
   const string  file_name       // имя файла
   )
```

Параметры

file_name

[in] Имя файла для удаления.

Примечание

Рабочая папка определяется ранее установленным/сброшенным при помощи метода SetCommon() флагом.

© 2000-2025, MetaQuotes Ltd.

IsExist

Проверяет существование файла.

```
bool  IsExist(
   const string  file_name      // имя файла
   )
```

Параметры

file_name

[in] Имя проверяемого файла.

Возвращаемое значение

true, если указанный файл существует.

© 2000-2025, MetaQuotes Ltd.

Copy

Копирует файл.

```
bool  Copy(
   const string  src_name,      // имя файла
   int           src_flag,      // флаг
   const string  dst_name,      // имя файла
   int           dst_flags      // флаги
   )
```

Параметры

src_name

[in] Имя файла-источника.

src_flag

[in] Флаги файла-источника (используется только FILE_COMMON).

dst_name

[in] Имя файла-приемника.

dst_flags

[in] Флаги файла-приемника (используются только FILE_REWRITE и FILE_COMMON).

Возвращаемое значение

true - в случае удачи, false - если не удалось скопировать файл.

© 2000-2025, MetaQuotes Ltd.

Move

Перемещает/переименовывает файл.

```
bool  Move(
   const string  src_name,       // имя файла
   int           src_flag,       // флаг
   const string  dst_name,       // имя файла
   int           dst_flags       // флаги
   )
```

Параметры

src_name

[in] Имя файла-источника.

src_flag

[in] Флаги файла-источника (используется только FILE_COMMON).

dst_name

[in] Имя файла-приемника.

dst_flags

[in] Флаги файла-приемника (используются только FILE_REWRITE и FILE_COMMON).

Возвращаемое значение

true - в случае удачи, false - если не удалось переместить/переименовать файл.

© 2000-2025, MetaQuotes Ltd.

Size

Получает размер файла в байтах.

```
ulong  Size()
```

Возвращаемое значение

размер файла в байтах. Если нет привязанного файла возвращается ULONG_MAX.

© 2000-2025, MetaQuotes Ltd.

Tell

Получает текущее положение файлового указателя.

```
ulong  Tell()
```

Возвращаемое значение

текущее положение файлового указателя. Если нет привязанного файла возвращается ULONG_MAX.

© 2000-2025, MetaQuotes Ltd.

Seek

Устанавливает положение файлового указателя.

```
void  Seek(
   long                offset,      // смещение
   ENUM_FILE_POSITION  origin       // точка отсчета
   )
```

Параметры

offset

[in] Смещение в байтах (может принимать и отрицательное значение).

origin

[in] Точка отсчета для смещения.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить файловый указатель.

© 2000-2025, MetaQuotes Ltd.

Flush

Сбрасывает на диск все данных, оставшиеся в файловом буфере ввода-вывода.

```
void  Flush()
```

© 2000-2025, MetaQuotes Ltd.

IsEnding

Определяет конец файла в процессе чтения.

```
bool  IsEnding()
```

Возвращаемое значение

true в случае, если в процессе чтения или перемещения файлового указателя достигнут конец файла.

© 2000-2025, MetaQuotes Ltd.

IsLineEnding

Определяет конец строки в текстовом файле в процессе чтения.

```
bool  IsLineEnding()
```

Возвращаемое значение

true в случае, если в процессе чтения txt или csv-файла достигнут конец строки (символы CR-LF).

© 2000-2025, MetaQuotes Ltd.

FolderCreate

Создает новую папку.

```
bool  FolderCreate(
   const string  folder_name      // имя папки
   )
```

Параметры

folder_name

[in] Имя папки, которую требуется создать. Содержит путь к папке (относительно рабочей папки, заданной при помощи флага FILE_COMMON).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать папку.

Примечание

Рабочая папка определяется ранее установленным/сброшенным при помощи метода SetCommon() флагом.

© 2000-2025, MetaQuotes Ltd.

FolderDelete

Удаляет указанную папку.

```
bool  FolderDelete(
   const string  folder_name      // имя папки
   )
```

Параметры

folder_name

[in] Имя папки, которую требуется удалить. Содержит путь к папке (относительно рабочей папки, заданной при помощи флага FILE_COMMON).

Возвращаемое значение

true - в случае удачи, false - если не удалось удалить папку.

Примечание

Рабочая папка определяется ранее установленным/сброшенным при помощи метода SetCommon() флагом.

© 2000-2025, MetaQuotes Ltd.

FolderClean

Очищает указанную папку.

```
bool  FolderClean(
   const string  folder_name      // имя папки
   )
```

Параметры

folder_name

[in] Имя папки, которую требуется очистить. Содержит путь к папке (относительно рабочей папки, заданной при помощи флага FILE_COMMON).

Возвращаемое значение

true - в случае удачи, false - если не удалось очистить папку.

Примечание

Рабочая папка определяется ранее установленным/сброшенным при помощи метода SetCommon() флагом.

© 2000-2025, MetaQuotes Ltd.

FileFindFirst

Начинает поиск файлов в соответствии с указанным фильтром.

```
int  FileFindFirst(
   const string  filter,         // фильтр поиска
   string&       file_name       // ссылка
   )
```

Параметры

filter

[in] Фильтр поиска.

file_name

[out] Ссылка на строку, куда в случае удачи помещается имя первого найденного файла.

Возвращаемое значение

Хэндл, который необходимо использовать для дальнейшего поиска файлов методом FileFindNext(), либо INVALID_HANDLE в случае, когда нет ни одного файла, соответствующего фильтру.

Примечание

Рабочая папка определяется ранее установленным/сброшенным при помощи метода SetCommon() флагом.

© 2000-2025, MetaQuotes Ltd.

FileFindNext

Продолжает поиск, начатый функцией FileFindFirst().

```
bool  FileFindNext(
   int      search_handle,     // хэндл поиска
   string&  file_name          // ссылка
   )
```

Параметры

search_handle

[in] Хэндл поиска, полученный методом FileFindFirst().

file_name

[in] Ссылка на строку, куда в случае удачи помещается имя найденного файла.

Возвращаемое значение

true - в случае удачи, false - если больше нет ни одного файла, соответствующего фильтру.

© 2000-2025, MetaQuotes Ltd.

FileFindClose

Закрывает хэндл поиска.

```
void  FileFindClose(
   int  search_handle      // хэндл поиска
   )
```

Параметры

search_handle

[in] Хэндл поиска, полученный методом FileFindFirst().

© 2000-2025, MetaQuotes Ltd.

Класс CFileBin

Класс CFileBin является классом для упрощенного доступа к двоичным файлам.

Описание

Класс CFileBin обеспечивает доступ к двоичным файлам.

Декларация

```
class CFileBin: public CFile
```

Заголовок

```
#include <Files\FileBin.mqh>
```

Иерархия наследования

CObject
  CFile
    CFileBin

Методы класса по группам

Открытие	
Open	Открывает двоичный файл
Методы записи	
WriteChar	Записывает переменную типа char или uchar
WriteShort	Записывает переменную типа short или ushort
WriteInteger	Записывает переменную типа int или uint
WriteLong	Записывает переменную типа long или ulong
WriteFloat	Записывает переменную типа float
WriteDouble	Записывает переменную типа double
WriteString	Записывает переменную типа string
WriteCharArray	Записывает массив переменных типа char или uchar
WriteShortArray	Записывает массив переменных типа short или ushort
WriteIntegerArray	Записывает массив переменных типа int или uint
WriteLongArray	Записывает массив переменных типа long или ulong
WriteFloatArray	Записывает массив переменных типа float
WriteDoubleArray	Записывает массив переменных типа double
WriteObject	Записывает данные экземпляра наследника класса CObject

© 2000-2025, MetaQuotes Ltd.

Открытие	
Методы чтения	
ReadChar	Читает переменную типа char или uchar
ReadShort	Читает переменную типа short или ushort
ReadInteger	Читает переменную типа int или uint
ReadLong	Читает переменную типа long или ulong
ReadFloat	Читает переменную типа float
ReadDouble	Читает переменную типа double
ReadString	Читает переменную типа string
ReadCharArray	Читает массив переменных типа char или uchar
ReadShortArray	Читает массив переменных типа short или ushort
ReadIntegerArray	Читает массив переменных типа int или uint
ReadLongArray	Читает массив переменных типа long или ulong
ReadFloatArray	Читает массив переменных типа float
ReadDoubleArray	Читает массив переменных типа double
ReadObject	Читает данные экземпляра наследника класса CObject

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Open

Открывает указанный двоичный файл и, в случае удачного открытия, привязывает его к экземпляру класса.

```
int  Open(
   const string  file_name,      // имя файла
   int           flags           // флаги
   )
```

Параметры

file_name

[in] Имя открываемого файла.

flags

[in] Флаги открытия файла (флаг FILE_BIN устанавливается принудительно).

Возвращаемое значение

Хэндл открытого файла.

© 2000-2025, MetaQuotes Ltd.

WriteChar

Записывает в файл переменную типа char или uchar.

```
uint  WriteChar(
   char  value      // значение
   )
```

Параметры

value

[in] Переменная для записи.

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteShort

Записывает в файл переменную типа short или ushort.

```
uint  WriteShort(
   short  value      // значение
   )
```

Параметры

value

[in] Переменная для записи.

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteInteger

Записывает в файл переменную типа int или uint.

```
uint  WriteInteger(
   int  value      // значение
   )
```

Параметры

value

[in] Переменная для записи.

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteLong

Записывает в файл переменную типа long или ulong.

```
uint  WriteLong(
   long  value      // значение
   )
```

Параметры

value

[in] Переменная для записи.

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteFloat

Записывает в файл переменную типа float.

```
uint  WriteFloat(
   float  value      // значение
   )
```

Параметры

value

[in] Переменная для записи.

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteDouble

Записывает в файл переменную типа double.

```
uint  WriteDouble(
   double  value      // значение
   )
```

Параметры

value

[in] Переменная для записи.

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteString

Записывает в файл переменную типа string.

```
uint  WriteString(
   const string  value      // значение
   )
```

Параметры

value

[in] Строка для записи.

Возвращаемое значение

Количество записанных байтов.

WriteString

Записывает в файл переменную типа string.

```
uint  WriteString(
   const string  value,     // значение
   int           size       // размер
   )
```

Параметры

value

[in] Строка для записи.

size

[in] Количество символов для записи.

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteCharArray

Записывает в файл массив переменных типа char или uchar.

```
uint  WriteCharArray(
   char&  array[],            // массив
   int    start_item=0,       // стартовый элемент
   int    items_count=-1      // количество элементов
   )
```

Параметры

array[]

[in] Массив для записи.

start_item=0

[in] Стартовый элемент для записи.

items_count=-1

[in] Количество элементов для записи (-1 - весь массив).

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteShortArray

Записывает в файл массив переменных типа short или ushort.

```
uint  WriteShortArray(
   short&  array[],            // массив
   int     start_item=0,       // стартовый элемент
   int     items_count=-1      // количество элементов
   )
```

Параметры

array[]

[in] Массив для записи.

start_item=0

[in] Стартовый элемент для записи.

items_count=-1

[in] Количество элементов для записи (-1 - весь массив).

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteIntegerArray

Записывает в файл массив переменных типа int или uint.

```
uint  WriteIntegerArray(
   int&  array[],             // массив
   int   start_item=0,        // стартовый элемент
   int   items_count=-1       // количество элементов
   )
```

Параметры

array[]

[in] Массив для записи.

start_item=0

[in] Стартовый элемент для записи.

items_count=-1

[in] Количество элементов для записи (-1 - весь массив).

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteLongArray

Записывает в файл массив переменных типа long или ulong.

```
uint  WriteLongArray(
   long&  array[],            // массив
   int    start_item=0,       // стартовый элемент
   int    items_count=-1      // количество элементов
   )
```

Параметры

array[]

[in] Массив для записи.

start_item=0

[in] Стартовый элемент для записи.

items_count=-1

[in] Количество элементов для записи (-1 - весь массив).

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteFloatArray

Записывает в файл массив переменных типа float.

```
uint  WriteFloatArray(
   float&  array[],            // массив
   int     start_item=0,       // стартовый элемент
   int     items_count=-1      // количество элементов
   )
```

Параметры

array[]

[in] Массив для записи.

start_item=0

[in] Стартовый элемент для записи.

items_count=-1

[in] Количество элементов для записи (-1 - весь массив).

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteDoubleArray

Записывает в файл массив переменных типа double.

```
uint  WriteDoubleArray(
   double&  array[],             // массив
   int      start_item=0,        // стартовый элемент
   int      items_count=-1       // количество элементов
   )
```

Параметры

array[]

[in] Массив для записи.

start_item=0

[in] Стартовый элемент для записи.

items_count=-1

[in] Количество элементов для записи (-1 - весь массив).

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

WriteObject

Записывает в файл данные экземпляра наследника класса CObject.

```
bool  WriteObject(
   CObject*  object      // указатель
   )
```

Параметры

object

[in] Указатель на экземпляр наследника класса CObject для записи.

Возвращаемое значение

true - в случае удачи, false - если не удалось записать данные.

© 2000-2025, MetaQuotes Ltd.

ReadChar

Читает из файла переменную типа char или uchar.

```
bool  ReadChar(
   char&  value      // значение флага
   )
```

Параметры

value

[in] Ссылка на переменную для размещения прочитанных данных.

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadShort

Читает из файла переменную типа short или ushort.

```
bool  ReadShort(
   short&  value
   )
```

Параметры

value

[in] Ссылка на переменную для размещения прочитанных данных.

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadInteger

Читает из файла переменную типа int или uint.

```
bool  ReadInteger(
   int&  value      // переменная
   )
```

Параметры

value

[in] Ссылка на переменную для размещения прочитанных данных.

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadLong

Читает из файла переменную типа long или ulong.

```
bool  ReadLong(
   long&  value
   )
```

Параметры

value

[in] Ссылка на переменную для размещения прочитанных данных.

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadFloat

Читает из файла переменную типа float.

```
bool  ReadFloat(
   float&  value      // переменная
   )
```

Параметры

value

[in] Ссылка на переменную для размещения прочитанных данных.

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadDouble

Читает из файла переменную типа double.

```
bool  ReadDouble(
   double&  value
   )
```

Параметры

value

[in] Ссылка на переменную для размещения прочитанных данных.

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadString

Читает из файла переменную типа string.

```
bool  ReadString(
   string&  value      // строка
   )
```

Параметры

value

[in] Ссылка на строку для размещения прочитанных данных.

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

ReadString

Читает из файла переменную типа string.

```
bool  ReadString(
   string&  value
   )
```

Параметры

value

[in] Ссылка на строку для размещения прочитанных данных.

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadCharArray

Читает из файла массив переменных типа char или uchar.

```
bool  ReadCharArray(
   char&  array[],            // массив
   int    start_item=0,       // стартовый элемент
   int    items_count=-1      // количество элементов
   )
```

Параметры

array[]

[in] Ссылка на массив для размещения прочитанных данных.

start_item=0

[in] Стартовый элемент для чтения.

items_count=-1

[in] Количество элементов для чтения (-1 - до конца файла).

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadShortArray

Читает из файла массив переменных типа short или ushort.

```
bool  ReadShortArray(
   short&  array[],            // массив
   int     start_item=0,       // стартовый элемент
   int     items_count=-1      // количество элементов
   )
```

Параметры

array[]

[in] Ссылка на массив для размещения прочитанных данных.

start_item=0

[in] Стартовый элемент для чтения.

items_count=-1

[in] Количество элементов для чтения (-1 - конца файла).

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadIntegerArray

Читает из файла массив переменных типа int или uint.

```
bool  ReadIntegerArray(
   int&  array[],             // массив
   int   start_item=0,        // стартовый элемент
   int   items_count=-1       // количество элементов
   )
```

Параметры

array[]

[in] Ссылка на массив для размещения прочитанных данных.

start_item=0

[in] Стартовый элемент для чтения.

items_count=-1

[in] Количество элементов для чтения (-1 - конца файла).

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadLongArray

Читает из файла массив переменных типа long или ulong.

```
bool  ReadLongArray(
   long&  array[],            // массив
   int    start_item=0,       // стартовый элемент
   int    items_count=-1      // количество элементов
   )
```

Параметры

array[]

[in] Ссылка на массив для размещения прочитанных данных.

start_item=0

[in] Стартовый элемент для чтения.

items_count=-1

[in] Количество элементов для чтения (-1 - конца файла).

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadFloatArray

Читает из файла массив переменных типа float.

```
bool  ReadFloatArray(
   float&  array[],            // массив
   int     start_item=0,       // стартовый элемент
   int     items_count=-1      // количество элементов
   )
```

Параметры

array[]

[in] Ссылка на массив для размещения прочитанных данных.

start_item=0

[in] Стартовый элемент для чтения.

items_count=-1

[in] Количество элементов для чтения (-1 - конца файла).

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadDoubleArray

Читает из файла массив переменных типа double.

```
bool  ReadDoubleArray(
   double&  array[],             // массив
   int      start_item=0,        // стартовый элемент
   int      items_count=-1       // количество элементов
   )
```

Параметры

array[]

[in] Ссылка на массив для размещения прочитанных данных.

start_item=0

[in] Стартовый элемент для чтения.

items_count=-1

[in] Количество элементов для чтения (-1 - конца файла).

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

ReadObject

Читает из файла данные экземпляра наследника класса CObject.

```
bool  ReadObject(
   CObject*  object      // указатель
   )
```

Параметры

object

[in] Указатель на экземпляр наследника класса CObject для чтения.

Возвращаемое значение

true - в случае удачи, false - если не удалось прочитать данные.

© 2000-2025, MetaQuotes Ltd.

Класс CFileTxt

Класс CFileTxt является классом для упрощенного доступа к текстовым файлам.

Описание

Класс CFileTxt обеспечивает доступ к текстовым файлам.

Декларация

```
class CFileTxt: public CFile
```

Заголовок

```
#include <Files\FileTxt.mqh>
```

Иерархия наследования

CObject
  CFile
    CFileTxt

Методы класса по группам

Открытие	
Open	Открывает текстовый файл
Методы записи	
WriteString	Записывает переменную типа string
Методы чтения	
ReadString	Читает переменную типа string

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Open

Открывает указанный текстовый файл и, в случае удачного открытия, привязывает его к экземпляру класса.

```
int  Open(
   const string  file_name,      // имя файла
   int           flags           // флаги
   )
```

Параметры

file_name

[in] Имя открываемого файла.

flags

[in] Флаги открытия файла (флаг FILE_TXT устанавливается принудительно).

Возвращаемое значение

Хэндл открытого файла.

© 2000-2025, MetaQuotes Ltd.

WriteString

Записывает в файл переменную типа string.

```
uint  WriteString(
   const string  value      // строка
   )
```

Параметры

value

[in] Строка для записи.

Возвращаемое значение

Количество записанных байтов.

© 2000-2025, MetaQuotes Ltd.

ReadString

Читает из файла переменную типа string.

```
string  ReadString()
```

Возвращаемое значение

Считанная строка.

© 2000-2025, MetaQuotes Ltd.

Операции со строками

Этот раздел содержит технические детали работы с классом строковых операций и описание соответствующих компонентов стандартной библиотеки MQL5.

Использование класса строковых операций позволит сэкономить время при создании пользовательских приложений обрабатывающих текстовую информацию.

Стандартная библиотека MQL5 (в части работы с строками) размещается в рабочем каталоге терминала в папке Include\Strings.

Имя класса	Назначение класса
CString	Класс строковых операций

© 2000-2025, MetaQuotes Ltd.

Класс CString

Класс CString является классом для упрощенного доступа к переменным типа string.

Описание

Класс CString обеспечивает упрощенный доступ к функциям API MQL5 по работе с переменными типа string.

Декларация

```
class CString: public CObject
```

Заголовок

```
#include <Strings\String.mqh>
```

Иерархия наследования

CObject

CString

Методы класса по группам

Методы доступа к данным	
Str	Получает строку
Len	Получает длину строки
Copy	Получает копию строки
Методы наполнения	
Fill	Заполняет строку
Assign	Присваивает строку
Append	Добавляет строку
Insert	Вставляет строку
Методы сравнения	
Compare	Сравнивает строки
CompareNoCase	Сравнивает строки без учета регистра
Методы работы с подстроками	
Left	Получает подстроку слева
Right	Получает подстроку справа
Mid	Получает подстроку из середины

© 2000-2025, MetaQuotes Ltd.

Методы доступа к данным	
Методы обрезания/удаления	
Trim	Обрезает строку слева и справа
TrimLeft	Обрезает строку слева
TrimRight	Обрезает строку справа
Clear	Очищает строку
Методы преобразования	
ToUpper	Переводит строку в верхний регистр
ToLower	Переводит строку в нижний регистр
Reverse	Разворачивает строку
Методы поиска	
Find	Ищет подстроку слева-направо
FindRev	Ищет подстроку справа-налево
Remove	Удаляет подстроку
Replace	Заменяет подстроку

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type

© 2000-2025, MetaQuotes Ltd.

Str

Получает строку.

```
string  Str() const;
```

Возвращаемое значение

Копия строки.

© 2000-2025, MetaQuotes Ltd.

Len

Получает длину строки.

```
uint  Len() const;
```

Возвращаемое значение

Длина строки.

© 2000-2025, MetaQuotes Ltd.

Copy

Копирует строку по ссылке.

```
void  Copy(
   string&  copy      // ссылка
   ) const;
```

Параметры

copy

[in] Ссылка на строку для данных.

Copy

Копирует строку в экземпляр класса CString.

```
void  Copy(
   CString*  copy      // указатель
   ) const;
```

Параметры

copy

[in] Указатель на экземпляр класса CString для данных.

© 2000-2025, MetaQuotes Ltd.

Fill

Заполняет строку указанным символом.

```
bool  Fill(
   short  character      // символ
   )
```

Параметры

character

[in] Символ для заполнения строки.

Возвращаемое значение

true - в случае удачи, false - если не удалось заполнить строку.

© 2000-2025, MetaQuotes Ltd.

Assign

Присваивает строку.

```
void  Assign(
   const string  str      // строка
   )
```

Параметры

str

[in] Строка для присвоения.

Assign

Присваивает строку из экземпляра класса CString.

```
void  Assign(
   CString*  str      // указатель
   )
```

Параметры

str

[in] Указатель на экземпляр класса CString для присвоения.

© 2000-2025, MetaQuotes Ltd.

Append

Добавляет строку.

```
void  Append(
   const string  str      // строка
   )
```

Параметры

str

[in] Строка для добавления.

Append

Добавляет строку из экземпляра класса CString.

```
void  Append(
   CString*  string      // указатель
   )
```

Параметры

string

[in] Указатель на экземпляр класса CString для добавления.

© 2000-2025, MetaQuotes Ltd.

Insert

Вставляет строку в указанную позицию.

```
uint  Insert(
   uint           pos,      // позиция
   const string   str       // строка
   )
```

Параметры

pos

[in] Позиция для вставки.

str

[in] Строка для вставки.

Возвращаемое значение

Длина результирующей строки.

Insert

Вставляет строку в указанную позицию из экземпляра класса CString.

```
uint  Insert(
   uint       pos,      // позиция
   CString*   str       // указатель
   )
```

Параметры

pos

[in] Позиция для вставки.

str

[in] Указатель на экземпляр класса CString для вставки.

Возвращаемое значение

Длина результирующей строки.

© 2000-2025, MetaQuotes Ltd.

Compare

Сравнивает со строкой.

```
int  Compare(
   const string  str      // строка
   ) const;
```

Параметры

str

[in] Строка для сравнения.

Возвращаемое значение

0 - если строки равны, -1 - если строка-член класса меньше чем строка-параметр, 1 - если строка-член класса больше чем строка-параметр.

Compare

Сравнивает со строкой экземпляра класса CString.

```
int  Compare(
   CString*  str      // указатель
   ) const;
```

Параметры

str

[in] Указатель на экземпляр класса CString для сравнения.

Возвращаемое значение

0 - если строки равны, -1 - если строка-член класса меньше чем строка-параметр, 1 - если строка-член класса больше чем строка-параметр.

© 2000-2025, MetaQuotes Ltd.

CompareNoCase

Сравнивает со строкой без учета регистра.

```
int  CompareNoCase(
   const string  str      // строка
   ) const;
```

Параметры

str

[in] Строка для сравнения.

Возвращаемое значение

0 - если строки равны, -1 - если строка-член класса меньше чем строка-параметр, 1 - если строка-член класса больше чем строка-параметр.

CompareNoCase

Сравнивает со строкой экземпляра класса CString без учета регистра.

```
int  CompareNoCase(
   CString*  str      // указатель
   ) const;
```

Параметры

str

[in] Указатель на экземпляр класса CString для сравнения.

Возвращаемое значение

0 - если строки равны, -1 - если строка-член класса меньше чем строка-параметр, 1 - если строка-член класса больше чем строка-параметр.

© 2000-2025, MetaQuotes Ltd.

Left

Получает подстроку указанной длины с начала строки.

```
string  Left(
   uint  count       // длина
   )
```

Параметры

count

[in] Длина подстроки.

Возвращаемое значение

Результирующая подстрока.

© 2000-2025, MetaQuotes Ltd.

Right

Получает подстроку указанной длины с конца строки.

```
string  Right(
   uint  count       // длина
   )
```

Параметры

count

[in] Длина подстроки.

Возвращаемое значение

Результирующая подстрока.

© 2000-2025, MetaQuotes Ltd.

Mid

Получает подстроку указанной длины с указанной позиции строки.

```
string  Mid(
   uint  pos,        // позиция
   uint  count       // длина
   )
```

Параметры

pos

[in] Позиция подстроки.

count

[in] Длина подстроки.

Возвращаемое значение

Результирующая подстрока.

© 2000-2025, MetaQuotes Ltd.

Trim

Удаляет из строки все символы входящие в набор (дополнительно ' ','\t','\r','\n') в начале и в конце строки.

```
int  Trim(
   const string  targets      // набор
   )
```

Параметры

targets

[in] Набор символов для удаления.

Возвращаемое значение

Количество удаленных символов.

Пример:

```
//--- 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());
  }
```

© 2000-2025, MetaQuotes Ltd.

TrimLeft

Удаляет из строки все символы входящие в набор (дополнительно ' ','\t','\r','\n') в начале строки.

```
int  TrimLeft(
   const string  targets      // набор
   )
```

Параметры

targets

[in] Набор символов для удаления.

Возвращаемое значение

Количество удаленных символов.

© 2000-2025, MetaQuotes Ltd.

TrimRight

Удаляет из строки все символы входящие в набор (дополнительно ' ','\t','\r','\n') в конце строки.

```
int  TrimRight(
   const string  targets      // набор
   )
```

Параметры

targets

[in] Набор символов для удаления.

Возвращаемое значение

Количество удаленных символов.

© 2000-2025, MetaQuotes Ltd.

Clear

Очищает строку

```
bool  Clear()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось очистить строку.

© 2000-2025, MetaQuotes Ltd.

ToUpper

Переводит все символы строки в верхний регистр.

```
bool  ToUpper()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить регистр.

© 2000-2025, MetaQuotes Ltd.

ToLower

Переводит все символы строки в нижний регистр.

```
bool  ToLower()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить регистр.

© 2000-2025, MetaQuotes Ltd.

Reverse

Разворачивает строку (меняет местами попарно начальные и конечные символы).

```
void  Reverse()
```

© 2000-2025, MetaQuotes Ltd.

Find

Ищет первое включение подстроки с указанной позиции.

```
int  Find(
   uint           start,          // позиция
   const string   substring       // подстрока
   ) const;
```

Параметры

start

[in] Стартовая позиция поиска подстроки.

substring

[in] Образец подстроки для поиска.

Возвращаемое значение

Индекс первого включения подстроки (-1 если подстрока не найдена).

© 2000-2025, MetaQuotes Ltd.

FindRev

Ищет последнее включение подстроки.

```
int  FindRev(
   const string  substring      // подстрока
   ) const;
```

Параметры

substring

[in] Образец подстроки для поиска.

Возвращаемое значение

Индекс последнего включения подстроки (-1 если подстрока не найдена).

© 2000-2025, MetaQuotes Ltd.

Remove

Удаляет все включения подстроки.

```
uint  Remove(
   const string  substring      // подстрока
   )
```

Параметры

substring

[in] Образец подстроки для поиска.

Возвращаемое значение

Количество удалений подстроки.

© 2000-2025, MetaQuotes Ltd.

Replace

Заменяет все включения подстроки.

```
uint  Replace(
   const string  substring,      // подстрока
   const string  newstring       // подстрока
   )
```

Параметры

substring

[in] Образец подстроки для поиска.

newstring

[in] Образец подстроки для замены.

Возвращаемое значение

Количество замен подстроки.

© 2000-2025, MetaQuotes Ltd.

Графические объекты

Этот раздел содержит технические детали работы с классами графических объектов и описание соответствующих компонентов стандартной библиотеки MQL5.

Использование классов графических объектов позволит сэкономить время при создании пользовательских программ (скриптов, экспертов).

Стандартная библиотека MQL5 (в части графических объектов) размещается в рабочем каталоге терминала в папке Include\ChartObjects.

Класс/группа	Описание
Базовый класс для работы с графическим объектом CChartObject	Базовый класс графического объекта
Линии	Группа классов "Линии"
Каналы	Группа классов "Каналы"
Инструменты Ганна	Группа классов "Ганн"
Инструменты Фибоначчи	Группа классов "Фибоначчи"
Инструменты Эллиотта	Группа классов "Эллиотт"
Фигуры	Группа классов "Фигуры"
Стрелки	Группа классов "Стрелки"
Элементы управления	Группа классов "Управление"

© 2000-2025, MetaQuotes Ltd.

CChartObject

Класс CChartObject является базовым классом для классов графических объектов чарта стандартной библиотеки MQL5.

Описание

Класс CChartObject обеспечивает всем своим потомкам возможность упрощенного доступа к функциям API MQL5.

Декларация

```
class CChartObject : public CObject
```

Заголовок

```
#include <ChartObjects\ChartObject.mqh>
```

Иерархия наследования

CObject

CChartObject

Прямые потомки

CChartObjectArrow, CChartObjectBitmap, CChartObjectBmpLabel, CChartObjectCycles, CChartObjectElliottWave3, CChartObjectEllipse, CChartObjectFiboArc, CChartObjectFiboFan, CChartObjectFiboTimes, CChartObjectHLine, CChartObjectRectangle, CChartObjectSubChart, CChartObjectText, CChartObjectTrend, CChartObjectTriangle, CChartObjectVLine

Методы класса по группам

Атрибуты	
ChartId	Получает идентификатор чарта, которому принадлежит графический объект
Window	Получает номер окна чарта в котором находится графический объект
Name	Получает/устанавливает наименование графического объекта
NumPoints	Получает количество точек привязки
Наполнение	
Attach	Привязывает графический объект чарта
SetPoint	Устанавливает параметры точки привязки
Удаление	
Delete	Удаляет графический объект чарта

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
Detach	Отвязывает графический объект чарта
Перемещение	
ShiftObject	Относительное перемещение объекта
ShiftPoint	Относительное перемещение точки объекта
Свойства объекта	
Time	Получает/устанавливает временную координату точки объекта
Price	Получает/устанавливает ценовую координату точки объекта
Color	Получает/устанавливает цвет объекта
Style	Получает/устанавливает стиль линии объекта
Width	Получает/устанавливает ширину линии объекта
Background	Получает/устанавливает флаг рисования объекта фоном
Selected	Получает/устанавливает флаг выбранности объекта
Selectable	Получает/устанавливает флаг выбираемости объекта
Description	Получает/устанавливает текст объекта
Tooltip	Получает/устанавливает текст всплывающей подказки объекта
Timeframes	Получает/устанавливает маску флагов видимости объекта
Z_Order	Получает/устанавливает приоритет графического объекта на получение события нажатия мышки на графике
CreateTime	Получает время создания объекта
Свойства уровней объекта	
LevelsCount	Получает/устанавливает количество уровней объекта
LevelColor	Получает/устанавливает цвет линии уровня
LevelStyle	Получает/устанавливает стиль линии уровня
LevelWidth	Получает/устанавливает ширину линии уровня
LevelValue	Получает/устанавливает значение уровня
LevelDescription	Получает/устанавливает текст уровня
Доступ к функциям API MQL5	
GetInteger	Получает значение свойства объекта
SetInteger	Устанавливает значение свойства объекта

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
GetDouble	Получает значение свойства объекта
SetDouble	Устанавливает значение свойства объекта
GetString	Получает значение свойства объекта
SetString	Устанавливает значение свойства объекта
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

© 2000-2025, MetaQuotes Ltd.

ChartId

Получает идентификатор чарта, которому принадлежит графический объект.

```
long  ChartId() const
```

Возвращаемое значение

Идентификатор чарта, на котором находится графический объект. Если нет привязанного объекта, возвращается -1.

Пример:

```
//--- example for CChartObject::ChartId
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get chart idintifier of chart object
   long chart_id=object.ChartId();
  }
```

© 2000-2025, MetaQuotes Ltd.

Window

Получает номер окна чарта в котором находится графический объект.

```
int  Window() const
```

Возвращаемое значение

Номер окна чарта в котором находится графический объект (0 - основное окно). Если нет привязанного объекта, возвращается -1.

Пример:

```
//--- example for CChartObject::Window
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get window of chart object
   int window=object.Window();
  }
```

© 2000-2025, MetaQuotes Ltd.

Name (метод Get)

Получает наименование графического объекта.

```
string  Name() const
```

Возвращаемое значение

Наименование графического объекта привязанного к экземпляру класса. Если нет привязанного объекта возвращается NULL.

Name (метод Set)

Устанавливает наименование графического объекта.

```
bool  Name(
   string  name      // новое наименование
   )
```

Параметры

name

[in] Новое наименование графического объекта.

Возвращаемое значение

true в случае успеха, false - если не удалось изменить наименование.

Пример:

```
//--- 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");
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

NumPoints

Получает количество точек привязки графического объекта.

```
int  NumPoints() const
```

Возвращаемое значение

Количество точек привязки графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Пример:

```
//--- example for CChartObject::NumPoints
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get points count of chart object
   int points=object.NumPoints();
  }
```

© 2000-2025, MetaQuotes Ltd.

Attach

Привязывает графический объект к экземпляру класса.

```
bool  Attach(
   long     chart_id,      // идентификатор чарта
   string   name,          // имя объекта
   int      window,        // окно чарта
   int      points         // количество точек
   )
```

Параметры

chart_id

[out] Иидентификатор чарта.

name

[in] Наименование (имя) графического объекта.

window

[in] Номер окна чарта (0 - основное окно).

points

[in] Количество точек привязки графического объекта.

Возвращаемое значение

true - в случае удачи, false - если нет возможности привязать объект.

Пример:

```
//--- 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;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SetPoint

Устанавливает новые координаты указанной точки привязки графического объекта.

```
bool  SetPoint(
   int       point,          // номер точки
   datetime  new_time,       // координата времени
   double    new_price       // координата цены
   )
```

Параметры

point

[in] Номер точки привязки графического объекта.

new_time

[in] Новое значение координаты времени указанной точки привязки

new_price

[in] Новое значение координаты цены указанной точки привязки.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить координаты точки.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет привязанный графический объект с чарта.

```
bool  Delete()
```

Возвращаемое значение

true - в случае удачи, false - если нет возможности удалить объект.

Пример:

```
//--- example for CChartObject::Delete
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- detach chart object
   if(!object.Delete())
     {
      printf("Object delete error");
      return;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Detach

Отвязывает графический объект.

```
void  Detach()
```

Возвращаемое значение

Нет.

Пример:

```
//--- example for CChartObject::Detach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- detach chart object
   object.Detach();
  }
```

© 2000-2025, MetaQuotes Ltd.

ShiftObject

Перемещает графический объект.

```
bool  ShiftObject(
   datetime  d_time,       // приращение координаты времени
   double    d_price       // приращение координаты цены
   )
```

Параметры

d_time

[in] Приращение координаты времени всех точек привязки

d_price

[in] Приращение координаты цены всех точек привязки.

Возвращаемое значение

true - в случае удачи, false - если не удалось переместить объект.

Пример:

```
//--- 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

ShiftPoint

Перемещает указанную точку привязки графического объекта.

```
bool  ShiftPoint(
   int       point,        // номер точки
   datetime  d_time,       // приращение координаты времени
   double    d_price       // приращение координаты цены
   )
```

Параметры

point

[in] Номер точки привязки графического объекта.

d_time

[in] Приращение координаты времени указанной точки привязки.

d_price

[in] Приращение координаты цены указанной точки привязки.

Возвращаемое значение

true - в случае удачи, false - если не удалось переместить точку.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Time (метод Get)

Получает координату времени указанной точки привязки графического объекта.

```
datetime  Time(
   int  point      // номер точки
   ) const
```

Параметры

point

[in] Номер точки привязки графического объекта.

Возвращаемое значение

Координату времени указанной точки привязки графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, либо у объекта нет указанной точки, возвращается 0.

Time (метод Set)

Устанавливает координату времени указанной точки привязки графического объекта.

```
bool  Time(
   int       point,         // номер точки
   datetime  new_time       // время
   )
```

Параметры

point

[in] Номер точки привязки графического объекта.

new_time

[in] Новое значение координаты времени указанной точки привязки графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить координату времени.

Пример:

```
//--- example for CChartObject::Time
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- get time of the chart object point
      datetime point_time=object.Time(i);
```

© 2000-2025, MetaQuotes Ltd.

```
      if(point_time==0)
        {
         //--- set time of the chart object point
         object.Time(i,TimeCurrent());
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Price (метод Get)

Получает координату цены указанной точки привязки графического объекта.

```
double  Price(
   int  point      // номер точки
   ) const
```

Параметры

point

[in] Номер точки привязки графического объекта.

Возвращаемое значение

Координата цены указанной точки привязки графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, либо у объекта нет указанной точки, возвращается EMPTY_VALUE.

Price (метод Set)

Устанавливает координату цены указанной точки привязки графического объекта.

```
bool  Price(
   int     point,          // номер точки
   double  new_price       // цена
   )
```

Параметры

point

[in] Номер точки привязки графического объекта.

new_price

[in] Новое значение координаты цены указанной точки привязки графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить координату цены.

Пример:

```
//--- 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 the chart object point
```

© 2000-2025, MetaQuotes Ltd.

```
      double point_price=object.Price(i);
      if(point_price!=price)
        {
         //--- set price for the chart object point
         object.Price(i,price);
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Color (метод Get)

Получает цвет линии графического объекта.

```
color  Color() const
```

Возвращаемое значение

Цвет линии графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта возвращается CLR_NONE.

Color (метод Set)

Устанавливает цвет линии графического объекта.

```
bool  Color(
   color  new_color      // новый цвет
   )
```

Параметры

new_color

[in] Новое значение цвета линии графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Style (метод Get)

Получает стиль линии графического объекта.

```
ENUM_LINE_STYLE  Style() const
```

Возвращаемое значение

Стиль линии графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается WRONG_VALUE.

Style (метод Set)

Устанавливает стиль линии графического объекта.

```
bool  Style(
   ENUM_LINE_STYLE  new_style      // стиль
   )
```

Параметры

new_style

[in] Новое значение стиля линии графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить стиль.

Пример:

```
//--- 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);
     }
  }
```

Смотри также

© 2000-2025, MetaQuotes Ltd.

Width (метод Get)

Получает толщину линии графического объекта.

```
int  Width() const
```

Возвращаемое значение

Толщину линии графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается -1.

Width (метод Set)

Устанавливает толщину линии графического объекта.

```
bool  Width(
   int  new_width      // толщина
   )
```

Параметры

new_width

[in] Новое значение толщины линии графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить толщину.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Background (метод Get)

Получает флаг рисования графического объекта на заднем плане.

```
bool  Background() const
```

Возвращаемое значение

Флаг рисования на заднем плане, графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта возвращается false.

Background (метод Set)

Устанавливает флаг рисования графического объекта на заднем плане.

```
bool  Background(
   bool  background      // значение флага
   )
```

Параметры

background

[in] Новое значение флага рисования графического объекта на заднем плане.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Selected (метод Get)

Получает флаг "выбранности" графического объекта. Иными словами - выбран графический объект или нет.

```
bool  Selected() const
```

Возвращаемое значение

Флаг "выбранности" графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

Selected (метод Set)

Устанавливает флаг "выбранности" графического объекта.

```
bool  Selected(
   bool  selected      // значение флага
   )
```

Параметры

selected

[in] Новое значение флага "выбранности" графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Selectable (метод Get)

Получает флаг "выбираемости" графического объекта. Иными словами - может ли графический объект быть выбран или нет.

```
bool  Selectable() const
```

Возвращаемое значение

Флаг "выбираемости" графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта возвращается false.

Selectable (метод Set)

Устанавливает флаг "выбираемости" графического объекта.

```
bool  Selectable(
   bool  selectable      // значение флага
   )
```

Параметры

selectable

[in] Новое значение флага "выбираемости" графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Description (метод Get)

Получает описание (текст) графического объекта.

```
string  Description() const
```

Возвращаемое значение

Описание (текст) графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается NULL.

Description (метод Set)

Устанавливает описание (текст) графического объекта.

```
bool  Description(
   string  text      // текст
   )
```

Параметры

text

[in] Новое значение описания (текста) графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить описание (текст).

Пример:

```
//--- 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");
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Tooltip (метод Get)

Получает текст всплывающей подсказки.

```
string  Tooltip() const
```

Возвращаемое значение

Текст всплывающей подсказки графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается NULL.

Tooltip (метод Set)

Устанавливает текст всплывающей подсказки.

```
bool  Tooltip(
   string  new_tooltip      // новый текст всплывающей подсказки
   )
```

Параметры

new_tooltip

[in] Новое значение текста всплывающей подсказки графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить текст подсказки.

Примечание:

Если свойство не задано, то показывается подсказка, автоматически сформированная терминалом. Показ подсказки можно отключить, установив для нее значение "\n" (перевод строки).

© 2000-2025, MetaQuotes Ltd.

Timeframes (метод Get)

Получает флаги видимости графического объекта.

```
int  Timeframes() const
```

Возвращаемое значение

Флаги видимости графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Timeframes (метод Set)

Устанавливает флаги видимости графического объекта.

```
bool  Timeframes(
   int  new_timeframes      // флаги видимости
   )
```

Параметры

new_timeframes

[in] Новые флаги видимости графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаги видимости.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Z_Order (метод Get)

Получает приоритет графического объекта на получение события нажатия мышки на графике (CHARTEVENT_CLICK).

```
long  Z_Order() const
```

Возвращаемое значение

Приоритет графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Z_Order (метод Set)

Устанавливает приоритет графического объекта на получение события нажатия мышки на графике (CHARTEVENT_CLICK).

```
bool  Z_Order(
   long  value      // приоритет графического объекта
   )
```

Параметры

value

[in] Новый приоритет графического объекта на получение события CHARTEVENT_CLICK.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить приоритет.

Примечание

Свойство Z_Order предназначено для управления приоритетом при отработке клика на графических объектах. Установив значение, больше значения по умолчанию (0), можно повысить приоритет объекта при обработке кликов мыши.

© 2000-2025, MetaQuotes Ltd.

CreateTime

Получает время создания графического объекта.

```
datetime  CreateTime() const
```

Возвращаемое значение

Время создания графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Пример:

```
//--- example for CChartObject::CreateTime
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get create time of chart object
   datetime create_time=object.CreateTime();
  }
```

© 2000-2025, MetaQuotes Ltd.

LevelsCount (метод Get)

Получает количество уровней графического объекта.

```
int  LevelsCount() const
```

Возвращаемое значение

Количество уровней графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

LevelsCount (метод Set)

Устанавливает количество уровней графического объекта.

```
bool  LevelsCount(
   int  levels      // количество уровней
   )
```

Параметры

levels

[in] Новое количество уровней графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить количество уровней.

Пример:

```
//--- 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

LevelColor (метод Get)

Получает цвет линии указанного уровня графического объекта.

```
color  LevelColor(
   int  level      // номер уровня
   ) const
```

Параметры

level

[in] Номер уровня графического объекта.

Возвращаемое значение

Цвет линии указанного уровня графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, либо у объекта нет указанного уровня, возвращается CLR_NONE.

LevelColor (метод Set)

Устанавливает цвет линии указанного уровня графического объекта.

```
bool  LevelColor(
   int    level,          // номер уровня
   color  new_color       // новый цвет
   )
```

Параметры

level

[in] Номер уровня графического объекта.

new_color

[in] Новое значение цвета линии указанного уровня графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

Пример:

```
//--- 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)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- set level color of chart object
          object.LevelColor(i,clrRed);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelStyle (метод Get)

Получает стиль линии указанного уровня графического объекта.

```
ENUM_LINE_STYLE  LevelStyle(
   int  level      // номер уровня
   ) const
```

Параметры

level

[in] Номер уровня графического объекта.

Возвращаемое значение

Стиль линии указанного уровня графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, либо у объекта нет указанного уровня, возвращается WRONG_VALUE.

LevelStyle (метод Set)

Устанавливает стиль линии указанного уровня графического объекта.

```
int  LevelStyle(
   int               level,     // номер уровня
   ENUM_LINE_STYLE   style      // стиль линии
   )
```

Параметры

level

[in] Номер уровня графического объекта.

style

[in] Новое значение стиля линии указанного уровня графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить стиль.

Пример:

```
//--- 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)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- set level style of chart object
          object.LevelStyle(i,STYLE_SOLID);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelWidth (метод Get)

Получает толщину линии указанного уровня графического объекта.

```
int  LevelWidth(
   int  level      // номер уровня
   ) const
```

Параметры

level

[in] Номер уровня графического объекта.

Возвращаемое значение

Толщину линии указанного уровня графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, либо у объекта нет указанного уровня, возвращается -1.

LevelWidth (метод Set)

Устанавливает толщину линии указанного уровня графического объекта.

```
bool  LevelWidth(
   int  level,          // номер уровня
   int  new_width       // новая толщина
   )
```

Параметры

level

[in] Номер уровня графического объекта.

new_width

[in] Новое значение толщины линии указанного уровня графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить толщину

Пример:

```
//--- 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)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- set level width of chart object
          object.LevelWidth(i,1);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelValue (метод Get)

Получает значение указанного уровня графического объекта.

```
double  LevelValue(
   int  level      // номер уровня
   ) const
```

Параметры

level

[in] Номер уровня графического объекта.

Возвращаемое значение

Значение указанного уровня графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, либо у объекта нет указанного уровня, возвращается EMPTY_VALUE.

LevelValue (метод Set)

Устанавливает значение указанного уровня графического объекта.

```
bool  LevelValue(
   int     level,          // номер уровня
   double  new_value       // новое значение
   )
```

Параметры

level

[in] Номер уровня графического объекта.

new_value

[in] Новое значение указанного уровня графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить значение уровня.

Пример:

```
//--- 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)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- set level value of chart object
          object.LevelValue(i,0.1*i);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelDescription (метод Get)

Получает описание (текст) указанного уровня графического объекта.

```
string  LevelDescription(
   int  level       // номер уровня
   ) const
```

Параметры

level

[in] Номер уровня графического объекта.

Возвращаемое значение

Описание (текст) указанного уровня графического объекта, привязанного к экземпляру класса. Если нет привязанного объекта, либо у объекта нет указанного уровня, возвращается NULL.

LevelDescription (метод Set)

Устанавливает описание (текст) указанного уровня графического объекта.

```
bool  LevelDescription(
   int     level,      // номер уровня
   string  text        // текст
   )
```

Параметры

level

[in] Номер уровня графического объекта.

text

[in] Новое значение описания (текста) указанного уровня графического объекта.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить описание (текст).

Пример:

```
//--- 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=="")
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- set level description of chart object
          object.LevelDescription(i,"Level_"+IntegerToString(i));
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

GetInteger

Обеспечивает упрощенный доступ к функциям API MQL5 ObjectGetInteger() для получения значений integer-свойств (имеющих типы bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) привязанного к экземпляру класса графического объекта. Существуют два варианта вызова функции:

Получение значения свойства без проверки корректности

```
long  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,          // идентификатор integer-свойства
   int                           modifier=-1       // модификатор
   ) const
```

Параметры

prop_id

[in] Идентификатор double-свойства графического объекта.

modifier=-1

[in] Модификатор (индекс) double-свойства.

Возвращаемое значение

Значение integer-свойства - в случае удачи, 0 - если нет возможности получить integer-свойство.

Получение значения свойства с проверкой корректности такого обращения

```
bool  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // идентификатор integer-свойства
   int                           modifier,      // модификатор
   long&                         value          // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор integer-свойства графического объекта.

modifier

[in] Модификатор (индекс) integer-свойства.

value

[out] Ссылка на переменную для размещения значения integer-свойства.

Возвращаемое значение

true - в случае удачи, false - если нет возможности получить integer-свойство.

Пример:

```
//--- example for CChartObject::GetInteger
#include <ChartObjects\ChartObject.mqh>
//---
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SetInteger

Обеспечивает упрощенный доступ к функциям API MQL5 ObjectSetInteger() для изменения integer-свойств (имеющих типы bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) привязанного к экземпляру класса графического объекта. Существуют два варианта вызова функции:

Установка значения свойства, не требующего модификатора

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // идентификатор integer-свойства
   long                           value         // значение
   )
```

Параметры

prop_id

[in] Идентификатор integer-свойства графического объекта.

value

[in] Новое значение изменяемого integer-свойства.

Установка значения свойства с указанием модификатора

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // идентификатор integer-свойства
   int                            modifier,     // модификатор
   long                           value         // значение
   )
```

Параметры

prop_id

[in] Идентификатор integer-свойства графического объекта.

modifier

[in] Модификатор (индекс) integer-свойства.

value

[in] Новое значение изменяемого integer-свойства.

Возвращаемое значение

true - в случае удачи, false - если нет возможности изменить integer-свойство.

Пример:

```
//--- example for CChartObject::SetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- set new color of chart object
   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;
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

GetDouble

Обеспечивает упрощенный доступ к функциям API MQL5 ObjectGetDouble() для получения значений double-свойств (имеющих типы float и double) привязанного к экземпляру класса графического объекта. Существуют два варианта вызова функции:

Получение значения свойства без проверки корректности

```
double  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,          // идентификатор double-свойства
   int                          modifier=-1      // модификатор
   ) const
```

Параметры

prop_id

[in] Идентификатор double-свойства графического объекта.

modifier=-1

[in] Модификатор (индекс) double-свойства.

Возвращаемое значение

Значение double-свойства - в случае удачи, EMPTY_VALUE - если нет возможности получить double-свойство.

Получение значения свойства с проверкой корректности такого обращения

```
bool  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,      // идентификатор double-свойства
   int                          modifier,     // модификатор
   double&                      value         // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор double-свойства графического объекта.

modifier

[in] Модификатор (индекс) double-свойства.

value

[out] Ссылка на переменную для размещения значения double-свойства.

Возвращаемое значение

true - в случае удачи, false - если нет возможности получить double-свойство.

Пример:

```
//--- example for CChartObject::GetDouble
#include <ChartObjects\ChartObject.mqh>
//---
```

© 2000-2025, 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SetDouble

Обеспечивает упрощенный доступ к функциям API MQL5 ObjectSetDouble() для изменения double-свойств (имеющих типы float и double) привязанного к экземпляру класса графического объекта. Существуют два варианта вызова функции:

Установка значения свойства, не требующего модификатора

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // идентификатор double-свойства
   double                        value         // значение
   )
```

Параметры

prop_id

[in] Идентификатор double-свойства графического объекта.

value

[in] Новое значение изменяемого double-свойства.

Установка значения свойства с указанием модификатора

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,       // идентификатор double-свойства
   int                           modifier,      // модификатор
   double                        value          // значение
   )
```

Параметры

prop_id

[in] Идентификатор double-свойства графического объекта.

modifier

[in] Модификатор (индекс) double-свойства.

value

[in] Новое значение изменяемого double-свойства.

Возвращаемое значение

true - в случае удачи, false - если нет возможности изменить double-свойство.

Пример:

```
//--- example for CChartObject::SetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
```

© 2000-2025, 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;
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

GetString

Обеспечивает упрощенный доступ к функциям API MQL5 ObjectGetString() для получения значений string-свойств, привязанного к экземпляру класса графического объекта. Существуют два варианта вызова функции:

Получение значения свойства без проверки корректности

```
string  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,          // идентификатор string-свойства
   int                          modifier=-1      // модификатор
   ) const
```

Параметры

prop_id

[in] Идентификатор string-свойства графического объекта.

modifier=-1

[in] Модификатор (индекс) string-свойства.

Возвращаемое значение

Значение string-свойства - в случае удачи, "" - если нет возможности получить string-свойство.

Получение значения свойства с проверкой корректности такого обращения

```
bool  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,       // идентификатор string-свойства
   int                          modifier,      // модификатор
   string&                      value          // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор string-свойства графического объекта.

modifier

[in] Модификатор (индекс) string-свойства.

value

[out] Ссылка на переменную для размещения значения string-свойства.

Возвращаемое значение

true - в случае удачи, false - если нет возможности получить string-свойство.

Пример:

```
//--- example for CChartObject::GetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000-2025, 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);
   }
}
```

© 2000-2025, MetaQuotes Ltd.

SetString

Обеспечивает упрощенный доступ к функциям API MQL5 ObjectSetString() для изменения string-свойств, привязанного к экземпляру класса графического объекта. Существуют два варианта вызова функции:

Установка значения свойства, не требующего модификатора

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // идентификатор string-свойства
   string                        value         // значение
   )
```

Параметры

prop_id

[in] Идентификатор string-свойства графического объекта.

value

[in] Новое значение изменяемого string-свойства.

Установка значения свойства с указанием модификатора

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,       // идентификатор string-свойства
   int                           modifier,      // модификатор
   string                        value          // значение
   )
```

Параметры

prop_id

[in] Идентификатор string-свойства графического объекта.

modifier

[in] Модификатор (индекс) string-свойства.

value

[in] Новое значение изменяемого string-свойства.

Возвращаемое значение

true - в случае удачи, false - если нет возможности изменить string-свойство.

Пример:

```
//--- example for CChartObject::SetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- set new name of chart object
```

© 2000-2025, 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;
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого при помощи функции FileOpen(...) файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого при помощи функции FileOpen(...) файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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
   //--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObject - 0x8888).

Пример:

```
//--- example for CChartObject::Type
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get object type
   int type=object.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

Объекты "Линии"

Группа графических объектов "Линии".

Этот раздел содержит технические детали работы с группой классов графических объектов "Линии" и описание соответствующих компонентов стандартной библиотеки MQL5.

Имя класса	Объект
CChartObjectVLine	Графический объект "Вертикальная линия"
CChartObjectHLine	Графический объект "Горизонтальная линия"
CChartObjectTrend	Графический объект "Трендовая линия"
CChartObjectTrendByAngle	Графический объект "Трендовая линия по углу"
CChartObjectCycles	Графический объект "Циклические линии"

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

CChartObjectVLine

Класс CChartObjectVLine является классом для упрощенного доступа к свойствам графического объекта "Вертикальная линия".

Описание

Класс CChartObjectVLine обеспечивает доступ к свойствам объекта "Вертикальная линия".

Декларация

```
class CChartObjectVLine : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectVLine

Методы класса по группам

Создание	
Create	Создает графический объект "Вертикальная линия"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Вертикальная линия".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time           // координата времени
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time

[in] Координата времени точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectVLine - OBJ_VLINE).

© 2000-2025, MetaQuotes Ltd.

CChartObjectHLine

Класс CChartObjectHLine является классом для упрощенного доступа к свойствам графического объекта "Горизонтальная линия".

Описание

Класс CChartObjectHLine обеспечивает доступ к свойствам объекта "Горизонтальная линия".

Декларация

```
class CChartObjectHLine : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectHLine

Методы класса по группам

Создание	
Create	Создает графический объект "Горизонтальная линия"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Горизонтальная линия".

```
bool  Create(
   long    chart_id,      // идентификатор чарта
   string  name,          // имя объекта
   long    window,        // окно чарта
   double  price          // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

price

[in] Координата цены точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectHLine - OBJ_HLINE).

© 2000-2025, MetaQuotes Ltd.

CChartObjectTrend

Класс CChartObjectTrend является классом для упрощенного доступа к свойствам графического объекта "Трендовая линия".

Описание

Класс CChartObjectTrend обеспечивает доступ к свойствам объекта "Трендовая линия".

Декларация

```
class CChartObjectTrend : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Иерархия наследования

CObject

CChartObject

CChartObjectTrend

Прямые потомки

CChartObjectChannel, CChartObjectFibo, CChartObjectFiboChannel, CChartObjectFiboExpansion, CChartObjectGannFan, CChartObjectGannGrid, CChartObjectPitchfork, CChartObjectRegression, CChartObjectStdDevChannel, CChartObjectTrendByAngle

Методы класса по группам

Создание	
Create	Создает графический объект "Трендовая линия"
Свойства	
RayLeft	Получить/установить свойство "луч влево"
RayRight	Получить/установить свойство "луч вправо"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CChartObject

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Трендовая линия".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

RayLeft (метод Get)

Получает значение флага "Луч влево".

```
bool  RayLeft() const
```

Возвращаемое значение

Значение флага "Луч влево" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

RayLeft (метод Set)

Устанавливает значение флага "Луч влево".

```
bool  RayLeft(
   bool  ray      // значение флага
   )
```

Параметры

ray

[in] Новое значение флага "Луч влево".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

RayRight (метод Get)

Получает значение флага "Луч вправо".

```
bool  RayRight() const
```

Возвращаемое значение

Значение флага "Луч вправо" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

RayRight (метод Set)

Устанавливает значение флага "Луч вправо".

```
bool  RayRight(
   bool  ray      // значение флага
   )
```

Параметры

ray

[in] Новое значение флага "Луч влево".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого при помощи функции FileOpen(...) файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого при помощи функции FileOpen(...) файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectTrend - OBJ_TREND).

© 2000-2025, MetaQuotes Ltd.

CChartObjectTrendByAngle

Класс CChartObjectTrendByAngle является классом для упрощенного доступа к свойствам графического объекта "Трендовая линия по углу".

Описание

Класс CChartObjectTrendByAngle обеспечивает доступ к свойствам объекта "Трендовая линия по углу".

Декларация

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Иерархия наследования

CObject

CChartObject

CChartObjectTrend

CChartObjectTrendByAngle

Прямые потомки

CChartObjectGannLine

Методы класса по группам

Создание	
Create	Создает графический объект "Трендовая линия по углу"
Свойства	
Angle	Получить/установить свойство "угол"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Трендовая линия по углу".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   long      window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Angle (метод Get)

Получает значение свойства "Угол".

```
double  Angle() const
```

Возвращаемое значение

Значение свойства "Угол" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается EMPTY_VALUE.

Angle (метод Set)

Устанавливает значение свойства "Угол".

```
bool  Angle(
   double  angle      // угол
   )
```

Параметры

angle

[in] Новое значение свойства "Угол".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectTrendByAngle - OBJ_TRENDBYANGLE).

© 2000-2025, MetaQuotes Ltd.

CChartObjectCycles

Класс CChartObjectCycles является классом для упрощенного доступа к свойствам графического объекта "Циклические линии".

Описание

Класс CChartObjectCycles обеспечивает доступ к свойствам объекта "Циклические линии".

Декларация

```
class CChartObjectCycles : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectCycles

Методы класса по группам

Создание	
Create	Создает графический объект "Циклические линии"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Циклические линии".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   long      window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectCycles - OBJ_CYCLES).

© 2000-2025, MetaQuotes Ltd.

Объекты "Каналы"

Группа графических объектов "Каналы".

Этот раздел содержит технические детали работы с группой классов графических объектов "Каналы" и описание соответствующих компонентов стандартной библиотеки MQL5.

Имя класса	Объект
CChartObjectChannel	Графический объект "Равноудаленный канал"
CChartObjectRegression	Графический объект "Канал линейной регрессии"
CChartObjectStdDevChannel	Графический объект "Канал стандартных отклонений"
CChartObjectPitchfork	Графический объект "Вилы Эндрюса"

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

CChartObjectChannel

Класс CChartObjectChannel является классом для упрощенного доступа к свойствам графического объекта "Равноудаленный канал".

Описание

Класс CChartObjectChannel обеспечивает доступ к свойствам объекта "Равноудаленный канал".

Декларация

```
class CChartObjectChannel : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectChannel

Методы класса по группам

Создание	
Create	Создает графический объект "Равноудаленный канал"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Смотри также

© 2000-2025, MetaQuotes Ltd.

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Равноудаленный канал".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени первой точки привязки
   double    price1,        // координата цены первой точки привязки
   datetime  time2,         // координата времени второй точки привязки
   double    price2,        // координата цены второй точки привязки
   datetime  time3,         // координата времени третьей точки привязки
   double    price3         // координата цены третьей точки привязки
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

time3

[in] Координата времени третей точки привязки.

price3

[in] Координата цены третей точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectChannel - OBJ_CHANNEL).

© 2000-2025, MetaQuotes Ltd.

CChartObjectRegression

Класс CChartObjectRegression является классом для упрощенного доступа к свойствам графического объекта "Канал линейной регрессии".

Описание

Класс CChartObjectRegression обеспечивает доступ к свойствам объекта "Канал линейной регрессии".

Декларация

```
class CChartObjectRegression : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectRegression

Методы класса по группам

Создание	
Create	Создает графический объект "Канал линейной регрессии"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Смотри также

© 2000-2025, MetaQuotes Ltd.

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Канал линейной регрессии".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   long      window,        // окно чарта
   datetime  time1,         // координата времени
   datetime  time2          // координата времени
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

time2

[in] Координата времени второй точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectRegression - OBJ_REGRESSION).

© 2000-2025, MetaQuotes Ltd.

CChartObjectStdDevChannel

Класс CChartObjectStdDevChannel является классом для упрощенного доступа к свойствам графического объекта "Канал стандартных отклонений".

Описание

Класс CChartObjectStdDevChannel обеспечивает доступ к свойствам объекта "Канал стандартных отклонений".

Декларация

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectStdDevChannel

Методы класса по группам

Создание	
Create	Создает графический объект "Канал стандартных отклонений"
Свойства	
Deviations	Получает/устанавливает свойство "Отклонение"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Канал стандартных отклонений".

```
bool  Create(
   long      chart_id,       // идентификатор чарта
   string    name,           // имя объекта
   int       window,         // окно чарта
   datetime  time1,          // координата времени
   datetime  time2,          // координата времени
   double    deviation       // отклонение
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

time2

[in] Координата времени второй точки привязки.

deviation

[in] Значение свойства "Отклонение".

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Deviation (метод Get)

Получает значение свойства "Отклонение".

```
double  Deviation() const
```

Возвращаемое значение

Значение свойства "Отклонение" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается EMPTY_VALUE.

Deviation (метод Set)

Устанавливает значение свойства "Отклонение".

```
bool  Deviation(
   double  deviation      // отклонение
   )
```

Параметры

deviation

[in] Новое значение свойства "Отклонение".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого при помощи функции FileOpen(...) файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого при помощи функции FileOpen(...) файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectStdDevChannel - OBJ_STDDEVCHANNEL).

© 2000-2025, MetaQuotes Ltd.

CChartObjectPitchfork

Класс CChartObjectPitchfork является классом для упрощенного доступа к свойствам графического объекта "Вилы Эндрюса".

Описание

Класс CChartObjectPitchfork обеспечивает доступ к свойствам объекта "Вилы Эндрюса".

Декларация

```
class CChartObjectPitchfork : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectPitchfork

Методы класса по группам

Создание	
Create	Создает графический объект "Вилы Эндрюса"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Смотри также

© 2000-2025, MetaQuotes Ltd.

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Вилы Эндрюса".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   long      window,        // окно чарта
   datetime  time1,         // координата времени первой точки привязки
   double    price1,        // координата цены первой точки привязки
   datetime  time2,         // координата времени второй точки привязки
   double    price2,        // координата цены второй точки привязки
   datetime  time3,         // координата времени третьей точки привязки
   double    price3         // координата цены третьей точки привязки
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

time3

[in] Координата времени третей точки привязки.

price3

[in] Координата цены третей точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectPitchfork - OBJ_PITCHFORK).

© 2000-2025, MetaQuotes Ltd.

Инструменты Ганна

Группа графических объектов "Инструменты Ганна".

Этот раздел содержит технические детали работы с группой классов графических объектов "Инструменты Ганна" и описание соответствующих компонентов стандартной библиотеки MQL5.

Имя класса	Объект
CChartObjectGannLine	Графический объект "Линия Ганна"
CChartObjectGannFan	Графический объект "Веер Ганна"
CChartObjectGannGrid	Графический объект "Сетка Ганна"

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannLine

Класс CChartObjectGannLine является классом для упрощенного доступа к свойствам графического объекта "Линия Ганна".

Описание

Класс CChartObjectGannLine обеспечивает доступ к свойствам объекта "Линия Ганна".

Декларация

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

Заголовок

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectTrendByAngle
        CChartObjectGannLine

Методы класса по группам

Создание	
Create	Создает графический объект "Линия Ганна"
Свойства	
PipsPerBar	Получить/установить свойство "Масштаб"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Методы унаследованные от CChartObjectTrendByAngle

Angle, Angle, Create

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Линия Ганна".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    ppb            // масштаб
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

ppb

[in] Масштаб

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

PipsPerBar (метод Get)

Получает значение свойства "Масштаб".

```
double  PipsPerBar() const
```

Возвращаемое значение

Значение свойства "Масштаб" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается EMPTY_VALUE.

PipsPerBar (метод Set)

Устанавливает значение свойства "Масштаб".

```
bool  PipsPerBar(
   double  ppb       // масштаб
   )
```

Параметры

ppb

[in] Новое значение свойства "Масштаб".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectGannLine - OBJ_GANNLINE).

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannFan

Класс CChartObjectGannFan является классом для упрощенного доступа к свойствам графического объекта "Веер Ганна".

Описание

Класс CChartObjectGannFan обеспечивает доступ к свойствам объекта "Веер Ганна".

Декларация

```
class CChartObjectGannFan : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannFan

Методы класса по группам

Создание	
Create	Создает графический объект "Веер Ганна"
Свойства	
PipsPerBar	Получить/установить свойство "Масштаб"
Downtrend	Получить/установить свойство "Тренд вниз"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Веер Ганна".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    ppb            // масштаб
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

ppb

[in] Масштаб

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

PipsPerBar (метод Get)

Получает значение свойства "Масштаб".

```
double  PipsPerBar() const
```

Возвращаемое значение

Значение свойства "Масштаб" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается EMPTY_VALUE.

PipsPerBar (метод Set)

Устанавливает значение свойства "Масштаб".

```
bool  PipsPerBar(
   double  ppb       // масштаб
   )
```

Параметры

ppb

[in] Новое значение свойства "Масштаб".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Downtrend (метод Get)

Получает значение флага "Тренд вниз".

```
bool  Downtrend() const
```

Возвращаемое значение

Значение флага Тренд вниз" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

Downtrend (метод Set)

Устанавливает значение флага "Тренд вниз".

```
bool  Downtrend(
   bool  downtrend      // значение флага
   )
```

Параметры

downtrend

[in] Новое значение флага "Тренд вниз".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные элемента списка в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectGannFan - OBJ_GANNFAN).

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannGrid

Класс CChartObjectGannGrid является классом для упрощенного доступа к свойствам графического объекта "Сетка Ганна".

Описание

Класс CChartObjectGannGrid обеспечивает доступ к свойствам объекта "Сетка Ганна".

Декларация

```
class CChartObjectGannGrid : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannGrid

Методы класса по группам

Создание	
Create	Создает графический объект "Сетка Ганна"
Свойства	
PipsPerBar	Получить/установить свойство "Масштаб"
Downtrend	Получить/установить свойство "Тренд вниз"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Сетка Ганна".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    ppb            // масштаб
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

ppb

[in] Масштаб

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

PipsPerBar (метод Get)

Получает значение свойства "Масштаб".

```
double  PipsPerBar() const
```

Возвращаемое значение

Значение свойства "Масштаб" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается EMPTY_VALUE.

PipsPerBar (метод Set)

Устанавливает значение свойства "Масштаб".

```
bool  PipsPerBar(
   double  ppb       // масштаб
   )
```

Параметры

ppb

[in] Новое значение свойства "Масштаб".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Downtrend (метод Get)

Получает значение флага "Тренд вниз".

```
bool  Downtrend() const
```

Возвращаемое значение

Значение флага "Тренд вниз" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

Downtrend (метод Set)

Устанавливает значение флага "Тренд вниз".

```
bool  Downtrend(
   bool  downtrend      // значение флага
   )
```

Параметры

downtrend

[in] Новое значение флага "Тренд вниз".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectGannGrid - OBJ_GANNGRID).

© 2000-2025, MetaQuotes Ltd.

Инструменты Фибоначчи

Группа графических объектов "Инструменты Фибоначчи".

Этот раздел содержит технические детали работы с группой классов графических объектов "Инструменты Фибоначчи" и описание соответствующих компонентов стандартной библиотеки MQL5.

Имя класса	Объект
CChartObjectFibo	Графический объект "Линии Фибоначчи"
CChartObjectFiboTimes	Графический объект "Временные зоны Фибоначчи"
CChartObjectFiboFan	Графический объект "Веер Фибоначчи"
CChartObjectFiboArc	Графический объект "Дуги Фибоначчи"
CChartObjectFiboChannel	Графический объект "Канал Фибоначчи"
CChartObjectFiboExpansion	Графический объект "Расширение Фибоначчи"

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

CChartObjectFibo

Класс CChartObjectFibo является классом для упрощенного доступа к свойствам графического объекта "Линии Фибоначчи".

Описание

Класс CChartObjectFibo обеспечивает доступ к свойствам объекта "Линии Фибоначчи".

Декларация

```
class CChartObjectFibo : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFibo

Методы класса по группам

Создание	
Create	Создает графический объект "Линии Фибоначчи"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Смотри также

© 2000-2025, MetaQuotes Ltd.

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Линии Фибоначчи".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectFibo - OBJ_FIBO).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboTimes

Класс CChartObjectFiboTimes является классом для упрощенного доступа к свойствам графического объекта "Временные зоны Фибоначчи".

Описание

Класс CChartObjectFiboTimes обеспечивает доступ к свойствам объекта "Временные зоны Фибоначчи".

Декларация

```
class CChartObjectFiboTimes : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Иерархия наследования

CObject

CChartObject

CChartObjectFiboTimes

Методы класса по группам

Создание	
Create	Создает графический объект "Временные зоны Фибоначчи"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Временные зоны Фибоначчи".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectFiboTimes - OBJ_FIBOTIMES).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboFan

Класс CChartObjectFiboFan является классом для упрощенного доступа к свойствам графического объекта "Веер Фибоначчи".

Описание

Класс CChartObjectFiboFan обеспечивает доступ к свойствам объекта "Веер Фибоначчи".

Декларация

```
class CChartObjectFiboFan : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectFiboFan

Методы класса по группам

Создание	
Create	Создает графический объект "Веер Фибоначчи"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Веер Фибоначчи".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectFiboFan - OBJ_FIBOFAN).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboArc

Класс CChartObjectFiboArc является классом для упрощенного доступа к свойствам графического объекта "Дуги Фибоначчи".

Описание

Класс CChartObjectFiboArc обеспечивает доступ к свойствам объекта "Дуги Фибоначчи".

Декларация

```
class CChartObjectFiboArc : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectFiboArc

Методы класса по группам

Создание	
Create	Создает графический объект "Дуги Фибоначчи"
Свойства	
Scale	Получить/установить свойство "Масштаб"
Ellipse	Получить/установить свойство "Эллипс"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Дуги Фибоначчи".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2,        // координата цены
   double    scale          // масштаб
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

scale

[in] Масштаб.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Scale (метод Get)

Получает значение свойства "Масштаб".

```
double  Scale() const
```

Возвращаемое значение

Значение свойства "Масштаб" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается EMPTY_VALUE.

Scale (метод Set)

Устанавливает значение свойства "Масштаб".

```
bool  Scale(
   double  scale      // масштаб
   )
```

Параметры

scale

[in] Новое значение свойства "Масштаб".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Ellipse (метод Get)

Получает значение флага "Эллипс".

```
bool  Ellipse() const
```

Возвращаемое значение

Значение флага "Эллипс" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

Ellipse (метод Set)

Устанавливает значение флага "Эллипс".

```
bool  Ellipse(
   bool  ellipse      // значение флага
   )
```

Параметры

ellipse

[in] Новое значение флага "Эллипс".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectFiboArc - OBJ_FIBOARC).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboChannel

Класс CChartObjectFiboChannel является классом для упрощенного доступа к свойствам графического объекта "Канал Фибоначчи".

Описание

Класс CChartObjectFiboChannel обеспечивает доступ к свойствам объекта "Канал Фибоначчи".

Декларация

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboChannel

Методы класса по группам

Создание	
Create	Создает графический объект "Канал Фибоначчи"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Смотри также

© 2000-2025, MetaQuotes Ltd.

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Канал Фибоначчи".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2,        // координата цены
   datetime  time3,         // координата времени
   double    price3         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

time3

[in] Координата времени третей точки привязки.

price3

[in] Координата цены третей точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectFiboChannel - OBJ_FIBOCHANNEL).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboExpansion

Класс CChartObjectFiboExpansion является классом для упрощенного доступа к свойствам графического объекта "Расширение Фибоначчи".

Описание

Класс CChartObjectFiboExpansion обеспечивает доступ к свойствам объекта "Расширение Фибоначчи".

Декларация

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

Заголовок

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboExpansion

Методы класса по группам

Создание	
Create	Создает графический объект "Расширение Фибоначчи"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Смотри также

© 2000-2025, MetaQuotes Ltd.

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Расширение Фибоначчи".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2,        // координата 2-й цены
   datetime  time3,         // координата времени
   double    price3         // координата 3-й цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

time3

[in] Координата времени второй точки привязки.

price3

[in] Координата цены первой точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectFiboExpansion - OBJ_EXPANSION).

© 2000-2025, MetaQuotes Ltd.

Инструменты Эллиотта

Группа графических объектов "Инструменты Эллиотта".

Этот раздел содержит технические детали работы с группой классов графических объектов "Инструменты Эллиотта".

Имя класса	Объект
CChartObjectElliottWave3	Графический объект "Корректирующая волна"
CChartObjectElliottWave5	Графический объект "Импульсная волна"

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

CChartObjectElliottWave3

Класс CChartObjectElliottWave3 является классом для упрощенного доступа к свойствам графического объекта "Корректирующая волна".

Описание

Класс CChartObjectElliottWave3 обеспечивает доступ к свойствам объекта "Корректирующая волна".

Декларация

```
class CChartObjectElliottWave3 : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Иерархия наследования

CObject

CChartObject

CChartObjectElliottWave3

Прямые потомки

CChartObjectElliottWave5

Методы класса по группам

Создание	
Create	Создает графический объект "Корректирующая волна"
Свойства	
Degree	Получить/установить свойство "Степень"
Lines	Получить/установить свойство "Отображение линий"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Корректирующая волна" (3-волновка Эллиота).

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2,        // координата цены
   datetime  time3,         // координата времени
   double    price3         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

time3

[in] Координата времени третей точки привязки.

price3

[in] Координата цены третей точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Degree (метод Get)

Получает значение свойства "Способ маркировки волн".

```
ENUM_ELLIOT_WAVE_DEGREE  Degree() const
```

Возвращаемое значение

Значение свойства "Способ маркировки волн" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается WRONG_VALUE.

Degree (метод Set)

Устанавливает значение свойства "Способ маркировки волн".

```
bool  Degree(
   ENUM_ELLIOT_WAVE_DEGREE  degree      // значение свойства
   )
```

Параметры

degree

[in] Новое значение свойства "Способ маркировки волн".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Lines (метод Get)

Получает значение флага "Отображение линий".

```
bool  Lines() const
```

Возвращаемое значение

Значение флага "Отображение линий" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

Lines (метод Set)

Устанавливает значение флага "Отображение линий".

```
bool  Lines(
   bool  lines      // значение флага
   )
```

Параметры

lines

[in] Новое значение флага "Отображение линий".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные элемента списка в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectElliottWave3 - OBJ_ELLIOTWAVE3).

© 2000-2025, MetaQuotes Ltd.

CChartObjectElliottWave5

Класс CChartObjectElliottWave5 является классом для упрощенного доступа к свойствам графического объекта "Импульсная волна".

Описание

Класс CChartObjectElliottWave5 обеспечивает доступ к свойствам объекта "Импульсная волна".

Декларация

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

Заголовок

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectElliottWave3
      CChartObjectElliottWave5

Методы класса по группам

Создание	
Create	Создает графический объект "Импульсная волна"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectElliottWave3

Degree, Degree, Lines, Lines, Create, Save, Load

Смотри также

© 2000-2025, MetaQuotes Ltd.

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Импульсная волна" (5-волновка Эллиота").

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2,        // координата цены
   datetime  time3,         // координата времени
   double    price3,        // координата цены
   datetime  time4,         // координата времени
   double    price4,        // координата цены
   datetime  time5          // координата времени
   double    price5,        // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

time3

[in] Координата времени третей точки привязки.

price3

[in] Координата цены третей точки привязки.

time4

[in] Координата времени четвертой точки привязки.

© 2000-2025, MetaQuotes Ltd.

price4

[in] Координата цены четвертой точки привязки.

time5

[in] Координата времени пятой точки привязки.

price5

[in] Координата цены пятой точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectElliottWave5 - OBJ_ELLIOTWAVE5).

© 2000-2025, MetaQuotes Ltd.

Объекты "Фигуры"

Группа графических объектов "Фигуры".

Этот раздел содержит технические детали работы с группой классов графических объектов "Фигуры" и описание соответствующих компонентов стандартной библиотеки MQL5.

Имя класса	Объект
CChartObjectRectangle	Графический объект "Прямоугольник"
CChartObjectTriangle	Графический объект "Треугольник"
CChartObjectEllipse	Графический объект "Эллипс"

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

CChartObjectRectangle

Класс CChartObjectRectangle является классом для упрощенного доступа к свойствам графического объекта "Прямоугольник".

Описание

Класс CChartObjectRectangle обеспечивает доступ к свойствам объекта "Прямоугольник".

Декларация

```
class CChartObjectRectangle : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Иерархия наследования

CObject

CChartObject

CChartObjectRectangle

Методы класса по группам

Создание	
Create	Создает графический объект "Прямоугольник"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Прямоугольник".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   long      window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectRectangle - OBJ_RECTANGLE).

© 2000-2025, MetaQuotes Ltd.

CChartObjectTriangle

Класс CChartObjectTriangle является классом для упрощенного доступа к свойствам графического объекта "Треугольник".

Описание

Класс CChartObjectTriangle обеспечивает доступ к свойствам объекта "Треугольник".

Декларация

```
class CChartObjectTriangle : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Иерархия наследования

CObject

CChartObject

CChartObjectTriangle

Методы класса по группам

Создание	
Create	Создает графический объект "Треугольник"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Треугольник".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   long      window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2,        // координата цены
   datetime  time3,         // координата времени
   double    price3         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

time3

[in] Координата времени третей точки привязки.

price3

[in] Координата цены третей точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectTriangle - OBJ_TRIANGLE).

© 2000-2025, MetaQuotes Ltd.

CChartObjectEllipse

Класс CChartObjectEllipse является классом для упрощенного доступа к свойствам графического объекта "Эллипс".

Описание

Класс CChartObjectEllipse обеспечивает доступ к свойствам объекта "Эллипс".

Декларация

```
class CChartObjectEllipse : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectEllipse

Методы класса по группам

Создание	
Create	Создает графический объект "Эллипс"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Эллипс".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time1,         // координата времени
   double    price1,        // координата цены
   datetime  time2,         // координата времени
   double    price2,        // координата цены
   datetime  time3,         // координата времени
   double    price3         // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time1

[in] Координата времени первой точки привязки.

price1

[in] Координата цены первой точки привязки.

time2

[in] Координата времени второй точки привязки.

price2

[in] Координата цены второй точки привязки.

time3

[in] Координата времени третей точки привязки.

price3

[in] Координата цены третей точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectEllipse - OBJ_ELLIPSE).

© 2000-2025, MetaQuotes Ltd.

Объекты "Стрелки"

Группа графических объектов "Стрелки".

Этот раздел содержит технические детали работы с группой классов графических объектов "Стрелки" и описание соответствующих компонентов стандартной библиотеки MQL5. По существу, стрелка - это некоторый значок, отображаемый пользователю, которому сопоставляется определенный код. Существуют два вида графических объектов "Стрелка" для отображения значков на графиках:

- Объект "Стрелка", который позволяет указать код значка, отображаемый объектом.
- Группа объектов для отображения определенного вида значка (и соответствующего определенному фиксированному коду).

Класс для работы со стрелками, отображающими значки произвольного кода

Имя класса	Имя объекта-стрелки
CChartObjectArrow	"Стрелка"

Классы для работы со стрелками, отображающими значок фиксированного кода

Имя класса	Имя объекта-стрелки
CChartObjectArrowCheck	"Проверка" ("Галочка")
CChartObjectArrowDown	"Стрелка вниз"
CChartObjectArrowUp	"Стрелка вверх"
CChartObjectArrowStop	"Стоп"
CChartObjectArrowThumbUp	"Хорошо" ("Большой палец вверх")
CChartObjectArrowThumbDown	"Плохо" ("Большой палец вниз")
CChartObjectArrowLeftPrice	Левая ценовая метка
CChartObjectArrowRightPrice	Правая ценовая метка

Смотри также

Типы объектов, Способы привязки объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

CChartObjectArrow

Класс CChartObjectArrow является классом для упрощенного доступа к свойствам графических объектов "Стрелки".

Описание

Класс CChartObjectArrow обеспечивает всем своим потомкам доступ к общим свойствам объектов "Стрелки".

Декларация

```
class CChartObjectArrow : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

Иерархия наследования

CObject

CChartObject

CChartObjectArrow

Прямые потомки

CChartObjectArrowCheck, CChartObjectArrowDown, CChartObjectArrowLeftPrice, CChartObjectArrowRightPrice, CChartObjectArrowStop, CChartObjectArrowThumbDown, CChartObjectArrowThumbUp, CChartObjectArrowUp

Методы класса по группам

Создание	
Create	Создает графический объект "Стрелка"
Свойства	
ArrowCode	Получает/устанавливает код символа
Anchor	Получает/устанавливает тип привязки
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CChartObject

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Смотри также

Типы объектов, Способы привязки объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "стрелка".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time,          // время
   double    price,         // цена
   char      code           // код
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time

[in] Координата времени.

price

[in] Координата цены.

code

[in] Код "стрелки" (Wingdings).

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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());
      //---
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- use arrow
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrowCode (метод Get)

Получает код символа "стрелки".

```
char  ArrowCode() const
```

Возвращаемое значение

Код символа "стрелки" объекта, привязанного к экземпляру класса. Если нет привязанного объекта возвращается 0.

ArrowCode (метод Set)

Устанавливает код символа "стрелки".

```
bool  ArrowCode(
   char  code      // значение кода
   )
```

Параметры

code

[in] Новое значение кода "стрелки" (Wingdings).

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить код.

Пример:

```
//--- 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;
     }
//--- change the code of arrow
//--- . . .
//--- get code of arrow
   if(arrow.ArrowCode()!=code)
     {
     //--- set code of arrow
```

© 2000-2025, MetaQuotes Ltd.

```
      arrow.ArrowCode(code);
      }
//--- use arrow
//--- . . .
   }
```

© 2000-2025, MetaQuotes Ltd.

Anchor (метод Get)

Получает способ привязки объекта "стрелка" к чарту.

```
ENUM_ARROW_ANCHOR  Anchor() const
```

Возвращаемое значение

Способ привязки объекта "стрелка", привязанного к экземпляру класса, к чарту. Если нет привязанного объекта возвращается WRONG_VALUE.

Anchor (метод Set)

Устанавливает способ привязки объекта "стрелка" к чарту.

```
bool  Anchor(
   ENUM_ARROW_ANCHOR  anchor      // тип привязки
   )
```

Параметры

anchor

[in] Новое значение способа привязки.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить тип привязки.

Пример:

```
//--- 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);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- use arrow
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого при помощи функции FileOpen(...) файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] Хэндл ранее открытого при помощи функции FileOpen(...) файла.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectArrow - OBJ_ARROW).

Пример:

```
//--- example for CChartObjectArrow::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   //--- get arrow type
   int type=arrow.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

Стрелки с фиксированным кодом

Классы "Стрелки с фиксированным кодом" являются классами для упрощенного доступа к свойствам графических объектов:

Имя класса	Имя объекта-стрелки
CChartObjectArrowCheck	"Проверка" ("Галочка")
CChartObjectArrowDown	"Стрелка вниз"
CChartObjectArrowUp	"Стрелка вверх"
CChartObjectArrowStop	"Стоп"
CChartObjectArrowThumbUp	"Хорошо" ("Большой палец вверх")
CChartObjectArrowThumbDown	"Плохо" ("Большой палец вниз")
CChartObjectArrowLeftPrice	Левая ценовая метка
CChartObjectArrowRightPrice	Правая ценовая метка

Описание

Классы "Стрелки с фиксированным кодом" обеспечивают доступ к свойствам соответствующих объектов.

Декларации

```
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;
```

Заголовок

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

Методы класса по группам

Создание	
Create	Создает графический объект, соответствующий классу
Свойства	
ArrowCode	"Заглушка" для метода изменения кода символа
Ввод/вывод	

© 2000-2025, MetaQuotes Ltd.

Создание	
virtual Type	Виртуальный метод идентификации

Смотри также

Типы объектов, Способы привязки объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "стрелка с фиксированным кодом".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time,          // время
   double    price          // цена
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time

[in] Координата времени.

price

[in] Координата цены.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

Пример:

```
//--- 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))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- arrow create error
       printf("Arrow create: Error %d!",GetLastError());
       //---
       return;
      }
//--- use arrow
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrowCode

Запрещает изменение кода символа "стрелки".

```
bool  ArrowCode(
   char  code      // значение кода
   )
```

Параметры

code

[in] Любое значение.

Возвращаемое значение

Всегда false.

Пример:

```
//--- 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
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта:
CChartObjectArrowCheck - OBJ_ARROW_CHECK,
CChartObjectArrowDown - OBJ_ARROW_DOWN,
CChartObjectArrowUp - OBJ_ARROW_UP,
CChartObjectArrowStop - OBJ_ARROW_STOP,
CChartObjectArrowThumbDown - OBJ_ARROW_THUMB_DOWN,
CChartObjectArrowThumbUp - OBJ_ARROW_THUMB_UP,
CChartObjectArrowLeftPrice - OBJ_ARROW_LEFT_PRICE,
CChartObjectArrowRightPrice - OBJ_ARROW_RIGHT_PRICE.

Пример:

```
//--- 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();
  }
```

© 2000-2025, MetaQuotes Ltd.

Элементы управления

Группа графических объектов "Элементы управления".

Этот раздел содержит технические детали работы с группой классов графических объектов "Элементы управления" и описание соответствующих компонентов стандартной библиотеки MQL5.

Имя класса	Объект
CChartObjectText	Графический объект "Текст"
CChartObjectLabel	Графический объект "Текстовая метка"
CChartObjectEdit	Графический объект "Поле ввода"
CChartObjectButton	Графический объект "Кнопка"
CChartObjectSubChart	Графический объект "График"
CChartObjectBitmap	Графический объект "Рисунок"
CChartObjectBmpLabel	Графический объект "Графическая метка"
CChartObjectRectLabel	Графический объект "Прямоугольная метка"

Смотри также

Типы объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

CChartObjectText

Класс CChartObjectText является классом для упрощенного доступа к свойствам графического объекта "Текст".

Описание

Класс CChartObjectText обеспечивает доступ к свойствам объекта "Текст".

Декларация

```
class CChartObjectText : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Иерархия наследования

CObject

CChartObject

CChartObjectText

Прямые потомки

CChartObjectLabel

Методы класса по группам

Создание	
Create	Создает графический объект "Текст"
Свойства	
Angle	Получить/установить свойство "Угол"
Font	Получить/установить свойство "Шрифт"
FontSize	Получить/установить свойство "Размер шрифта"
Anchor	Получает/устанавливает свойство "Точка привязки"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CChartObject

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Прямые потомки класса:

- CChartObjectLabel

Смотри также

Типы объектов, Свойства объектов, Способы привязки объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Текст".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time,          // координата времени
   double    price          // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time

[in] Координата времени точки привязки.

price

[in] Координата цены точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Angle (метод Get)

Получает значение свойства "Угол наклона".

```
double  Angle() const
```

Возвращаемое значение

Значение свойства "Угол наклона" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается EMPTY_VALUE.

Angle (метод Set)

Устанавливает значение свойства "Угол наклона".

```
bool  Angle(
   double  angle      // значение свойства
   )
```

Параметры

angle

[in] Новое значение свойства "Угол наклона".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Font (метод Get)

Получает значение свойства "Шрифт".

```
string  Font() const
```

Возвращаемое значение

Значение свойства "Шрифт" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается "".

Font (метод Set)

Устанавливает значение свойства "Шрифт".

```
bool  Font(
   string  font      // значение свойства
   )
```

Параметры

font

[in] Новое значение свойства "Шрифт".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

FontSize (метод Get)

Получает значение свойства "Размер шрифта".

```
int  FontSize() const
```

Возвращаемое значение

Значение свойства "Размер шрифта" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

FontSize (метод Set)

Устанавливает значение свойства "Размер шрифта".

```
bool  FontSize(
   int  size      // значение свойства
   )
```

Параметры

size

[in] Новое значение свойства "Размер шрифта".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Anchor (метод Get)

Получает значение свойства "Положение точки привязки".

```
ENUM_ANCHOR_POINT  Anchor() const
```

Возвращаемое значение

Значение свойства "Положение точки привязки" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается WRONG_VALUE.

Anchor (метод Set)

Устанавливает значение свойства "Положение точки привязки".

```
bool  Anchor(
   ENUM_ANCHOR_POINT  anchor      // значение свойства
   )
```

Параметры

anchor

[in] Новое значение свойства "Положение точки привязки".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectText - OBJ_TEXT).

© 2000-2025, MetaQuotes Ltd.

CChartObjectLabel

Класс CChartObjectLabel является классом для упрощенного доступа к свойствам графического объекта "Текстовая метка".

Описание

Класс CChartObjectLabel обеспечивает доступ к свойствам объекта "Текстовая метка".

Декларация

```
class CChartObjectLabel : public CChartObjectText
```

Заголовок

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel

Прямые потомки

CChartObjectEdit, CChartObjectRectLabel

Методы класса по группам

Создание	
Create	Создает графический объект "Текстовая метка"
Свойства	
X_Distance	Получает/устанавливает свойство "Координата X"
Y_Distance	Получает/устанавливает свойство "Координата Y"
X_Size	Получает свойство "Размер по X"
Y_Size	Получает свойство "Размер по Y"
Corner	Получает/устанавливает свойство "Угол графика для привязки"
Time	"Заглушка" изменения координаты времени
Price	"Заглушка" изменения координаты цены
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла

© 2000-2025, MetaQuotes Ltd.

Создание	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Смотри также

Типы объектов, Свойства объектов, Способы привязки объектов, Угол привязки, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Текстовая метка".

```
bool  Create(
   long     chart_id,      // идентификатор чарта
   string   name,          // имя объекта
   int      window,        // окно чарта
   int      X,             // координата X
   int      Y             // координата Y
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

X

[in] Дистанция по оси X.

Y

[in] Дистанция по оси Y.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

X_Distance (метод Get)

Получает значение свойства "Дистанция по оси X".

```
int  X_Distance() const
```

Возвращаемое значение

Значение свойства "Дистанция по оси X" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

X_Distance (метод Set)

Устанавливает значение свойства "Дистанция по оси X".

```
bool  X_Distance(
   int  X      // значение свойства
   )
```

Параметры

X

[in] Новое значение свойства "Дистанция по оси X".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Y_Distance (метод Get)

Получает значение свойства "Дистанция по оси Y".

```
int  Y_Distance() const
```

Возвращаемое значение

Значение свойства "Дистанция по оси Y" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Y_Distance (метод Set)

Устанавливает значение свойства "Дистанция по оси Y".

```
bool  Y_Distance(
   int  Y      // значение свойства
   )
```

Параметры

Y

[in] Новое значение свойства "Дистанция по оси Y".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

X_Size

Получает значение свойства "Размер по оси X".

```
int  X_Size() const
```

Возвращаемое значение

Значение свойства "Размер по оси X" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Получает значение свойства "Размер по оси Y".

```
int  Y_Size() const
```

Возвращаемое значение

Значение свойства "Размер по оси Y" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

Corner (метод Get)

Получает значение свойства "Угол графика для привязки".

```
ENUM_BASE_CORNER  Corner() const
```

Возвращаемое значение

Значение свойства "Угол графика для привязки" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается WRONG_VALUE.

Corner (метод Set)

Устанавливает значение свойства "Угол графика для привязки".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // значение свойства
   )
```

Параметры

corner

[in] Новое значение свойства "Угол графика для привязки".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Time

Запрещает изменение координаты времени.

```
bool  Time(
   datetime  time      // любое значение
   )
```

Параметры

time

[in] Любое значение типа datetime.

Возвращаемое значение

всегда false.

© 2000-2025, MetaQuotes Ltd.

Price

Запрещает изменение координаты цены.

```
bool  Price(
   double  price      // любое значение
   )
```

Параметры

price

[in] Любое значение типа double.

Возвращаемое значение

всегда false.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectLabel - OBJ_LABEL).

© 2000-2025, MetaQuotes Ltd.

CChartObjectEdit

Класс CChartObjectEdit является классом для упрощенного доступа к свойствам графического объекта "Поле ввода".

Описание

Класс CChartObjectEdit обеспечивает доступ к свойствам объекта "Поле ввода".

Декларация

```
class CChartObjectEdit : public CChartObjectLabel
```

Заголовок

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectEdit

Прямые потомки

CChartObjectButton

Методы класса по группам

Создание	
Create	Создает графический объект "Поле ввода"
Свойства	
TextAlign	Получает/устанавливает свойство "Тип выравнивания текста"
X_Size	Получает свойство "Размер по X"
Y_Size	Получает свойство "Размер по Y"
BackColor	Получает/устанавливает свойство "Цвет фона"
BorderColor	Получает/устанавливает свойство "Цвет рамки"
ReadOnly	Получает/устанавливает свойство "Read Only"
Angle	"Заглушка" для изменений свойства "Угол"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла

© 2000-2025, MetaQuotes Ltd.

Создание	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Методы унаследованные от CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Смотри также

Типы объектов, Свойства объектов, Способы привязки объектов, Угол привязки, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Поле ввода".

```
bool  Create(
   long     chart_id,      // идентификатор чарта
   string   name,          // имя объекта
   int      window,        // окно чарта
   int      X,             // координата X
   int      Y,             // координата Y
   int      sizeX,         // размер X
   int      sizeY          // размер Y
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

X

[in] Дистанция по оси X.

Y

[in] Дистанция по оси Y.

sizeX

[in] Размер по оси X.

sizeY

[in] Размер по оси Y.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

TextAlign (метод Get)

Получает значение свойства "Тип выравнивания текста по горизонтали" графического объекта "Поле ввода".

```
ENUM_ALIGN_MODE  TextAlign() const
```

Возвращаемое значение

Значение свойства "Тип выравнивания текста по горизонтали" объекта, привязанного к экземпляру класса.

TextAlign (метод Set)

Устанавливает значение свойства "Тип выравнивания текста по горизонтали" графического объекта "Поле ввода".

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // значение свойства
   )
```

Параметры

align

[in] Новое значение свойства "Тип выравнивания текста по горизонтали".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

X_Size

Устанавливает значение свойства "Размер по оси X".

```
bool  X_Size(
   int  size      // значение свойства
   )
```

Параметры

size

[in] Новое значение свойства "Размер по оси X".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Устанавливает значение свойства "Размер по оси Y".

```
bool  Y_Size(
   int  size      // значение свойства
   )
```

Параметры

size

[in] Новое значение свойства "Размер по оси Y".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

BackColor (метод Get)

Получает значение свойства "Цвет фона".

```
color  BackColor() const
```

Возвращаемое значение

Значение свойства "Цвет фона" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается CLR_NONE.

BackColor (метод Set)

Устанавливает значение свойства "Цвет фона".

```
bool  BackColor(
   color  new_color      // значение свойства
   )
```

Параметры

new_color

[in] Новое значение свойства "Цвет фона".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

BorderColor (метод Get)

Получает значение свойства "Цвет рамки".

```
color  BorderColor() const
```

Возвращаемое значение

Значение свойства "Цвет рамки" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается CLR_NONE.

BorderColor (метод Set)

Устанавливает значение свойства "Цвет рамки".

```
bool  BorderColor(
   color  new_color      // новое значение свойства
   )
```

Параметры

new_color

[in] Новое значение свойства "Цвет рамки".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

ReadOnly (метод Get)

Получает значение свойства "Read Only".

```
bool  ReadOnly() const
```

Возвращаемое значение

Значение свойства "Read Only" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

ReadOnly (метод Set)

Устанавливает значение свойства "Read Only".

```
bool  ReadOnly(
   const  bool  flag      // новое значение свойства
   )
```

Параметры

flag

[in] Новое значение свойства "Read Only" (true означает запрет возможности редактирования текста).

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Angle

Запрещает изменение свойства "Угол наклона".

```
bool  Angle(
   double  angle      // любое значение
   )
```

Параметры

angle

[in] Любое значение типа double.

Возвращаемое значение

всегда false.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectEdit - OBJ_EDIT).

© 2000-2025, MetaQuotes Ltd.

CChartObjectButton

Класс CChartObjectButton является классом для упрощенного доступа к свойствам графического объекта "Кнопка".

Описание

Класс CChartObjectButton обеспечивает доступ к свойствам объекта "Кнопка".

Декларация

```
class CChartObjectButton : public CChartObjectEdit
```

Заголовок

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectEdit
          CChartObjectButton

Прямые потомки

CChartObjectPanel

Методы класса по группам

Создание	
Create	Унаследован от класса CChartObjectEdit
Свойства	
State	Получает/устанавливает состояние кнопки (Нажата/Отжата)
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CChartObject

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Методы унаследованные от CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Методы унаследованные от CChartObjectEdit

X_Size, Y_Size, BackColor, BackColor, BorderColor, BorderColor, ReadOnly, ReadOnly, TextAlign, TextAlign, Angle, Create

Смотри также

Типы объектов, Свойства объектов, Способы привязки объектов, Угол привязки, Графические объекты

© 2000-2025, MetaQuotes Ltd.

State (метод Get)

Получает значение свойства "Состояние".

```
bool  State() const
```

Возвращаемое значение

Значение свойства "Состояние" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

State (метод Set)

Устанавливает значение свойства "Состояние".

```
bool  State(
   bool  state      // значение свойства
   )
```

Параметры

state

[in] Новое значение свойства "Состояние".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectButton - OBJ_BUTTON).

© 2000-2025, MetaQuotes Ltd.

CChartObjectSubChart

Класс CChartObjectSubChart является классом для упрощенного доступа к свойствам графического объекта "График".

Описание

Класс CChartObjectSubChart обеспечивает доступ к свойствам объекта "График".

Декларация

```
class CChartObjectSubChart : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectSubChart

Методы класса по группам

Создание	
Create	Создает графический объект "График"
Свойства	
X_Distance	Получает/устанавливает свойство "Координата X"
Y_Distance	Получает/устанавливает свойство "Координата Y"
Corner	Получает/устанавливает свойство "Угол графика для привязки"
X_Size	Получает/устанавливает свойство "Размер по X"
Y_Size	Получает/устанавливает свойство "Размер по Y"
Symbol	Получает/устанавливает свойство "Символ"
Period	Получает/устанавливает свойство "Период"
Scale	Получает/устанавливает свойство "Масштаб"
DateScale	Получает/устанавливает свойство "Отображать шкалу времени"
PriceScale	Получает/устанавливает свойство "Отображать ценовую шкалу"
Time	"Заглушка" изменения координаты времени
Price	"Заглушка" изменения координаты цены
Ввод/вывод	

© 2000-2025, MetaQuotes Ltd.

Создание	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Свойства объектов, Угол привязки, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "График на графике".

```
bool  Create(
   long     chart_id,      // идентификатор чарта
   string   name,          // имя объекта
   int      window,        // окно чарта
   int      X,             // координата X
   int      Y,             // координата Y
   int      sizeX,         // размер X
   int      sizeY          // размер Y
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

X

[in] Дистанция по оси X.

Y

[in] Дистанция по оси Y.

sizeX

[in] Размер по оси X.

sizeY

[in] Размер по оси Y.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

X_Distance (метод Get)

Получает значение свойства "Дистанция по оси X".

```
int  X_Distance() const
```

Возвращаемое значение

Значение свойства "Дистанция по оси X" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

X_Distance (метод Set)

Устанавливает значение свойства "Дистанция по оси X".

```
bool  X_Distance(
   int  X      // значение свойства
   )
```

Параметры

X

[in] Новое значение свойства "Дистанция по оси X".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Y_Distance (метод Get)

Получает значение свойства "Дистанция по оси Y".

```
int  Y_Distance() const
```

Возвращаемое значение

Значение свойства "Дистанция по оси Y" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Y_Distance (метод Set)

Устанавливает значение свойства "Дистанция по оси Y".

```
bool  Y_Distance(
   int  Y      // значение свойства
   )
```

Параметры

Y

[in] Новое значение свойства "Дистанция по оси Y".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Corner (метод Get)

Получает значение свойства "Угол графика для привязки".

```
ENUM_BASE_CORNER  Corner() const
```

Возвращаемое значение

Значение свойства "Угол графика для привязки" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается WRONG_VALUE.

Corner (метод Set)

Устанавливает значение свойства "Угол графика для привязки".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // значение свойства
   )
```

Параметры

corner

[in] Новое значение свойства "Угол графика для привязки".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

X_Size (метод Get)

Получает значение свойства "Размер по оси X".

```
int  X_Size() const
```

Возвращаемое значение

Значение свойства "Размер по оси X" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

X_Size (метод Set)

Устанавливает значение свойства "Размер по оси X".

```
bool  X_Size(
   int  size      // значение свойства
   )
```

Параметры

size

[in] Новое значение свойства "Размер по оси X".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Y_Size (метод Get)

Получает значение свойства "Размер по оси Y".

```
int  Y_Size() const
```

Возвращаемое значение

Значение свойства "Размер по оси Y" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Y_Size (метод Set)

Устанавливает значение свойства "Размер по оси Y".

```
bool  Y_Size(
   int  size      // значение свойства
   )
```

Параметры

size

[in] Новое значение свойства "Размер по оси Y".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Symbol (метод Get)

Получает значение свойства "Символ".

```
string  Symbol() const
```

Возвращаемое значение

Значение свойства "Символ" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается "".

Symbol (метод Set)

Устанавливает значение свойства "Символ".

```
bool  Symbol(
   string  symbol       // символ
   )
```

Параметры

symbol

[in] Новое значение свойства "Символ".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Period (метод Get)

Получает значение свойства "Период".

```
int  Period() const
```

Возвращаемое значение

Значение свойства "Период" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Period (метод Set)

Устанавливает значение свойства "Период".

```
bool  Period(
   int  period      // период
   )
```

Параметры

period

[in] Новое значение свойства "Период".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Scale (метод Get)

Получает значение свойства "Масштаб".

```
double  Scale() const
```

Возвращаемое значение

Значение свойства "Масштаб" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается EMPTY_VALUE.

Scale (метод Set)

Устанавливает значение свойства "Масштаб".

```
bool  Scale(
   double  scale      // значение свойства
   )
```

Параметры

scale

[in] Новое значение свойства "Масштаб".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

DateScale (метод Get)

Получает значение флага "Отображение шкалы времени".

```
bool  DateScale() const
```

Возвращаемое значение

Значение флага "Отображение шкалы времени" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

DateScale (метод Set)

Устанавливает значение флага "Отображение шкалы времени".

```
bool  DateScale(
   bool  scale      // значение флага
   )
```

Параметры

scale

[in] Новое значение флага "Отображение шкалы времени".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

PriceScale (метод Get)

Получает значение флага "Отображение ценовой шкалы".

```
bool  1_PriceScale() const
```

Возвращаемое значение

Значение флага "Отображение ценовой шкалы" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

PriceScale (метод Set)

Устанавливает значение флага "Отображение ценовой шкалы".

```
bool  2_PriceScale(
   bool  scale      // значение флага
   )
```

Параметры

scale

[in] Новое значение флага "Отображение ценовой шкалы".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

Time

Запрещает изменение координаты времени.

```
bool  Time(
   datetime  time      // любое значение
   )
```

Параметры

time

[in]

Возвращаемое значение

всегда false.

© 2000-2025, MetaQuotes Ltd.

Price

Запрещает изменение координаты цены.

```
bool  Price(
   double  price      // любое значение
   )
```

Параметры

price

[in]

Возвращаемое значение

всегда false.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectSubChart - OBJ_CHART).

© 2000-2025, MetaQuotes Ltd.

CChartObjectBitmap

Класс CChartObjectBitmap является классом для упрощенного доступа к свойствам графического объекта "Рисунок".

Описание

Класс CChartObjectBitmap обеспечивает доступ к свойствам объекта "Рисунок".

Декларация

```
class CChartObjectBitmap : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectBitmap

Методы класса по группам

Создание	
Create	Создает графический объект "Рисунок"
Свойства	
BmpFile	Получает/устанавливает свойство "Имя BMP-файла"
X_Offset	Получает/устанавливает свойство "X-координата угла области видимости"
Y_Offset	Получает/устанавливает свойство "Y-координата угла области видимости"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

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

Смотри также

Типы объектов, Свойства объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Рисунок".

```
bool  Create(
   long      chart_id,      // идентификатор чарта
   string    name,          // имя объекта
   int       window,        // окно чарта
   datetime  time,          // координата времени
   double    price          // координата цены
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

time

[in] Координата времени точки привязки.

price

[in] Координата цены точки привязки.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

BmpFile (метод Get)

Получает значение свойства "Имя BMP-файла".

```
string  BmpFile() const
```

Возвращаемое значение

Значение свойства "Имя BMP-файла" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается "".

BmpFile (метод Set)

Устанавливает значение свойства "Имя BMP-файла".

```
bool  BmpFile(
   string  name      // значение свойства
   )
```

Параметры

name

[in] Новое значение свойства "Имя BMP-файла".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

X_Offset (метод Get)

Получает значение свойства "X-координата области видимости" (левого верхнего угла) графического объекта CChartObjectBitmap.

```
int  X_Offset() const
```

Возвращаемое значение

Значение свойства "X-координата области видимости" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

X_Offset (метод Set)

Устанавливает значение свойства "X-координата угла области видимости" (левого верхнего угла) графического объекта CChartObjectBitmap. Значение задается в пикселях относительного верхнего левого угла исходного изображения.

```
bool  X_Offset(
   int  X      // значение свойства
   )
```

Параметры

X

[in] Новое значение свойства "X-координата угла области видимости".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Y_Offset (метод Get)

Получает значение свойства "Y-координата области видимости" (левого верхнего угла) графического объекта CChartObjectBitmap.

```
int  Y_Offset() const
```

Возвращаемое значение

Значение свойства "Y-координата области видимости" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Y_Offset (метод Set)

Устанавливает значение свойства "Y-координата угла области видимости" (левого верхнего угла) графического объекта CChartObjectBitmap. Значение задается в пикселях относительного верхнего левого угла исходного изображения.

```
bool  Y_Offset(
   int  Y      // значение свойства
   )
```

Параметры

Y

[in] Новое значение свойства "Y-координата угла области видимости".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectBitmap - OBJ_BITMAP).

© 2000-2025, MetaQuotes Ltd.

CChartObjectBmpLabel

Класс CChartObjectBmpLabel является классом для упрощенного доступа к свойствам графического объекта "Графическая метка".

Описание

Класс CChartObjectBmpLabel обеспечивает доступ к свойствам объекта "Графическая метка".

Декларация

```
class CChartObjectBmpLabel : public CChartObject
```

Заголовок

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectBmpLabel

Методы класса по группам

Создание	
Create	Создает графический объект "Графическая метка"
Свойства	
X_Distance	Получает/устанавливает свойство "Координата X"
Y_Distance	Получает/устанавливает свойство "Координата Y"
X_Offset	Получает/устанавливает свойство "X-координата угла области видимости"
Y_Offset	Получает/устанавливает свойство "Y-координата угла области видимости"
Corner	Получает/устанавливает свойство "Угол графика для привязки"
X_Size	Получает свойство "Размер по X"
Y_Size	Получает свойство "Размер по Y"
BmpFileOn	Получает/устанавливает свойство "Имя BMP-файла" для состояние On (Нажата)
BmpFileOff	Получает/устанавливает свойство "Имя BMP-файла" для состояние Off (Отжата)
State	Получает/устанавливает состояние кнопки (Нажата/Отжата)
Time	"Заглушка" изменения координаты времени

© 2000-2025, MetaQuotes Ltd.

Создание	
Price	"Заглушка" изменения координаты цены
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Смотри также

Типы объектов, Свойства объектов, Угол привязки, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Графическая метка".

```
bool  Create(
   long     chart_id,      // идентификатор чарта
   string   name,          // имя объекта
   int      window,        // окно чарта
   int      X,             // координата X
   int      Y              // координата Y
   )
```

Параметры

chart_id

[in] Идентификатор чарта (0 - текущий чарт).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

X

[in] Дистанция по оси X.

Y

[in] Дистанция по оси Y.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

X_Distance (метод Get)

Получает значение свойства "Дистанция по оси X".

```
int  X_Distance() const
```

Возвращаемое значение

Значение свойства "Дистанция по оси X" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

X_Distance (метод Set)

Устанавливает значение свойства "Дистанция по оси X".

```
bool  X_Distance(
   int  X      // значение свойства
   )
```

Параметры

X

[in] Новое значение свойства "Дистанция по оси X".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Y_Distance (метод Get)

Получает значение свойства "Дистанция по оси Y".

```
int  Y_Distance() const
```

Возвращаемое значение

Значение свойства "Дистанция по оси Y" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Y_Distance (метод Set)

Устанавливает значение свойства "Дистанция по оси Y".

```
bool  Y_Distance(
   int  Y      // значение свойства
   )
```

Параметры

Y

[in] Новое значение свойства "Дистанция по оси Y".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

X_Offset (метод Get)

Получает значение свойства "X-координата области видимости" (левого верхнего угла) графического объекта CChartObjectBmpLabel.

```
int  X_Offset() const
```

Возвращаемое значение

Значение свойства "X-координата области видимости" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

X_Offset (метод Set)

Устанавливает значение свойства "X-координата угла области видимости" (левого верхнего угла) графического объекта CChartObjectBmpLabel. Значение задается в пикселях относительного верхнего левого угла исходного изображения.

```
bool  X_Offset(
   int  X      // значение свойства
   )
```

Параметры

X

[in] Новое значение свойства "X-координата угла области видимости".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Y_Offset (метод Get)

Получает значение свойства "Y-координата области видимости" (левого верхнего угла) графического объекта CChartObjectBmpLabel.

```
int  Y_Offset() const
```

Возвращаемое значение

Значение свойства "Y-координата области видимости" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

Y_Offset (метод Set)

Устанавливает значение свойства "Y-координата угла области видимости" (левого верхнего угла) графического объекта CChartObjectBmpLabel. Значение задается в пикселях относительного верхнего левого угла исходного изображения.

```
bool  Y_Offset(
   int  Y      // значение свойства
   )
```

Параметры

Y

[in] Новое значение свойства "Y-координата угла области видимости".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Corner (метод Get)

Получает значение свойства "Угол графика для привязки".

```
ENUM_BASE_CORNER  Corner() const
```

Возвращаемое значение

Значение свойства "Угол графика для привязки" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается WRONG_VALUE.

Corner (метод Set)

Устанавливает значение свойства "Угол графика для привязки".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // значение свойства
   )
```

Параметры

corner

[in] Новое значение свойства "Угол графика для привязки".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

X_Size

Получает значение свойства "Размер по оси X".

```
int  X_Size() const
```

Возвращаемое значение

Значение свойства "Размер по оси X" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Получает значение свойства "Размер по оси Y".

```
int  Y_Size() const
```

Возвращаемое значение

Значение свойства "Размер по оси Y" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

BmpFileOn (метод Get)

Получает значение свойства "Имя BMP-файла ON".

```
string  BmpFileOn() const
```

Возвращаемое значение

Значение свойства "Имя BMP-файла ON" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается "".

BmpFileOn (метод Set)

Устанавливает значение свойства "Имя BMP-файла ON".

```
bool  BmpFileOn(
   string  name      // имя файла
   )
```

Параметры

name

[in] Новое значение свойства "Имя BMP-файла ON".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

BmpFileOff (метод Get)

Получает значение свойства "Имя BMP-файла OFF".

```
string  BmpFileOff() const
```

Возвращаемое значение

Значение свойства "Имя BMP-файла OFF" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается "".

BmpFileOff (метод Set)

Устанавливает значение свойства "Имя BMP-файла OFF".

```
bool  BmpFileOff(
   string  name      // имя файла
   )
```

Параметры

name

[in] Новое значение свойства "Имя BMP-файла OFF".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

State (метод Get)

Получает значение свойства "Состояние".

```
bool  State() const
```

Возвращаемое значение

Значение свойства "Состояние" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается false.

State (метод Set)

Устанавливает значение свойства "Состояние".

```
bool  State(
   bool  state      // значение свойства
   )
```

Параметры

state

[in] Новое значение свойства "Состояние".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Time

Запрещает изменение координаты времени.

```
bool  Time(
   datetime  time      // любое значение
   )
```

Параметры

time

[in] Любое значение типа datetime.

Возвращаемое значение

всегда false.

© 2000-2025, MetaQuotes Ltd.

Price

Запрещает изменение координаты цены.

```
bool  Price(
   double  price      // любое значение
   )
```

Параметры

price

[in] Любое значение типа double.

Возвращаемое значение

всегда false.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectBmpLabel - OBJ_BITMAP_LABEL).

© 2000-2025, MetaQuotes Ltd.

CChartObjectRectLabel

Класс CChartObjectRectLabel является классом для упрощенного доступа к свойствам графического объекта "Прямоугольная метка".

Описание

Класс CChartObjectRectLabel обеспечивает доступ к свойствам объекта "Прямоугольная метка".

Декларация

```
class CChartObjectRectLabel : public CChartObjectLabel
```

Заголовок

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Иерархия наследования

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectRectLabel

Методы класса по группам

Создание	
Create	Создает графический объект "Прямоугольная метка"
Свойства	
X_Size	Устанавливает свойство "Размер по X"
Y_Size	Устанавливает свойство "Размер по Y"
BackColor	Получает/устанавливает свойство "Цвет фона"
Angle	Запрещает изменение свойства "Угол наклона"
BorderType	Получает/устанавливает свойство "Тип рамки"
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Методы унаследованные от CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Смотри также

Типы объектов, Свойства объектов, Графические объекты

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический объект "Прямоугольная метка".

```
bool  Create(
   long     chart_id,      // идентификатор графика
   string   name,          // имя объекта
   int      window,        // окно графика
   int      X,             // координата X
   int      Y,             // координата Y
   int      sizeX,         // размер X
   int      sizeY          // размер Y
   )
```

Параметры

chart_id

[in] Идентификатор графика (0 - текущий график).

name

[in] Уникальное имя создаваемого объекта.

window

[in] Номер окна чарта (0 - основное окно).

X

[in] Дистанция по оси X.

Y

[in] Дистанция по оси Y.

sizeX

[in] Размер по оси X.

sizeY

[in] Размер по оси Y.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

X_Size

Устанавливает значение свойства "Размер по оси X".

```
bool  X_Size(
   int  size      // значение свойства
   )
```

Параметры

size

[in] Новое значение свойства "Размер по оси X".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

Примечание

Получение значений свойств "Размер по оси X" и "Размер по оси Y" осуществляется при помощи методов X_Size и Y_Size родительского класса CChartObjectLabel.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Устанавливает значение свойства "Размер по оси Y".

```
bool  Y_Size(
   int  size      // значение свойства
   )
```

Параметры

size

[in] Новое значение свойства "Размер по оси Y".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

Примечание

Получение значений свойств "Размер по оси X" и "Размер по оси Y" осуществляется при помощи методов X_Size и Y_Size родительского класса CChartObjectLabel.

© 2000-2025, MetaQuotes Ltd.

BackColor

Получает значение свойства "Цвет фона".

```
color  BackColor() const
```

Возвращаемое значение

Значение свойства "Цвет фона" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

BackColor

Устанавливает значение свойства "Цвет фона".

```
bool  BackColor(
   color  new_color      // значение свойства
   )
```

Параметры

new_color

[in] Новое значение свойства "Цвет фона".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Angle

Запрещает изменение свойства "Угол наклона".

```
bool  Angle(
   double  angle      // любое значение
   )
```

Параметры

angle

[in] Любое значение типа double.

Возвращаемое значение

Всегда false.

© 2000-2025, MetaQuotes Ltd.

BorderType

Получает значение свойства "Тип рамки".

```
int  BorderType() const
```

Возвращаемое значение

Значение свойства "Тип рамки" объекта, привязанного к экземпляру класса. Если нет привязанного объекта, возвращается 0.

BorderType

Устанавливает значение свойства "Тип рамки".

```
bool  BorderType(
   int  type      // значение свойства
   )
```

Параметры

type

[in] Новое значение свойства "Тип рамки".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет параметры объекта в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл бинарного файла, ранее открытого при помощи функции FileOpen.

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа графического объекта.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа объекта (для CChartObjectRectLabel - OBJ_RECTANGLE_LABEL).

© 2000-2025, MetaQuotes Ltd.

Пользовательская графика

В этом разделе представлены инструменты для работы с пользовательской графикой. Их использование существенно упрощает построение пользовательских графиков, рисунков и визуализацию данных.

Отдельно разработаны классы для создания графических объектов и примитивов, для отрисовки различных видов диаграмм и кривых. Реализованы различные возможности отображения объектов: изменение стиля и цвета линий, заливка, работа с сериями данных на графике и т.д.

Для того чтобы начать работать с библиотекой, достаточно прочитать статью Визуализируй это! Графическая библиотека в MQL5 как аналог plot из R

Класс	Описание
CCanvas	Класс для упрощенного создания пользовательских рисунков
CChartCanvas	Базовый класс для реализации классов, предназначенных для отрисовки графиков и их элементов
CHistogramChart	Класс для построения столбчатых гистограмм
CLineChart	Класс для отрисовки кривых
CPieChart	Класс для построения круговых диаграмм

© 2000-2025, MetaQuotes Ltd.

CCanvas

Класс CCanvas является классом для упрощенного создания пользовательских рисунков.

Описание

Класс CCanvas обеспечивает создание графического ресурса (с привязкой к объекту чарта или без) и рисование графических примитивов.

Декларация

```
class CCanvas
```

Заголовок

```
#include <Canvas\Canvas.mqh>
```

Иерархия наследования

CCanvas

Прямые потомки

CChartCanvas, CFlameCanvas

Методы класса по группам

Создание	
Attach	Привязывает объект OBJ_BITMAP_LABEL к экземпляру класса CCanvas
Create	Создает графический ресурс без привязки к объекту чарта
CreateBitmap	Создает графический ресурс, привязанный к объекту чарта
CreateBitmapLabel	Создает графический ресурс, привязанный к объекту чарта
Destroy	Уничтожает графический ресурс
Свойства	
ChartObjectName	Получает имя привязанного объекта чарта
ResourceName	Получает имя графического ресурса
Width	Получает ширину графического ресурса
Height	Получает высоту графического ресурса
LineStyleSet	Устанавливает стиль линии
Обновление объекта на экране	
Update	Отображает изменения на экран

© 2000-2025, MetaQuotes Ltd.

Создание	
Resize	Изменяет размеры графического ресурса
Очистка/заполнение цветом	
Erase	Очищает или заполняет указанным цветом
Доступ к данным	
PixelGet	Получает цвет точки с указанными координатами
PixelSet	Устанавливает цвет точки с указанными координатами
Рисование примитивов	
LineVertical	Рисует вертикальную линию
LineHorizontal	Рисует горизонтальную линию
Line	Рисует произвольную линию
Polyline	Рисует ломаную линию
Polygon	Рисует многоугольник
Rectangle	Рисует прямоугольник
Circle	Рисует окружность
Triangle	Рисует треугольник
Ellipse	Рисует эллипс
Arc	Рисует дугу эллипса
Pie	Рисует сектор эллипса
Рисование закрашенных примитивов	
FillRectangle	Рисует закрашенный прямоугольник
FillCircle	Рисует закрашенный круг
FillTriangle	Рисует закрашенный треугольник
FillPolygon	Рисует закрашенный многоугольник
FillEllipse	Рисует закрашенный эллипс
Fill	Закрашивает область
Рисование примитивов с использованием сглаживания	
PixelSetAA	Рисует точку
LineAA	Рисует линию

© 2000-2025, MetaQuotes Ltd.

Создание	
PolylineAA	Рисует ломаную линию
PolygonAA	Рисует многоугольник
TriangleAA	Рисует треугольник
CircleAA	Рисует окружность
EllipseAA	Рисует эллипс
LineWu	Рисует линию
PolylineWu	Рисует ломаную линию
PolygonWu	Рисует многоугольник
TriangleWu	Рисует треугольник
CircleWu	Рисует окружность
EllipseWu	Рисует эллипс
LineThick	Рисует отрезок произвольной линии заданной толщины с использованием алгоритма сглаживания
LineThickVertical	Рисует вертикальный отрезок произвольной линии заданной толщины с использованием алгоритма сглаживания
LineThickHorizontal	Рисует горизонтальный отрезок произвольной линии заданной толщины с использованием алгоритма сглаживания
PolygonSmooth	Рисует многоугольник заданной толщины с использованием двух алгоритмов сглаживания
PolygonThick	Рисует многоугольник заданной толщины с использованием алгоритма сглаживания
PolylineSmooth	Рисует ломаную линию заданной толщины с использованием двух алгоритмов сглаживания
PolylineThick	Рисует ломаную линию заданной толщины с использованием алгоритма сглаживания
Текст	
FontSet	Устанавливает параметры шрифта
FontNameSet	Устанавливает имя шрифта
FontSizeSet	Устанавливает размер шрифта
FontFlagsSet	Устанавливает флаги шрифта
FontAngleSet	Устанавливает угол наклона шрифта
FontGet	Получает параметры шрифта
FontNameGet	Получает имя шрифта

© 2000-2025, MetaQuotes Ltd.

Создание	
FontSizeGet	Получает размер шрифта
FontFlagsGet	Получает флаги шрифта
FontAngleGet	Получает угол наклона шрифта
TextOut	Выводит текст
TextWidth	Получает ширину текста
TextHeight	Получает высоту текста
TextSize	Получает размеры текста
Прозрачность	
TransparentLevelSet	Устанавливает уровень прозрачности
Ввод/вывод	
LoadFromFile	Читает рисунок из BMP-файла

© 2000-2025, MetaQuotes Ltd.

Attach

Получает из объекта OBJ_BITMAP_LABEL графический ресурс и привязывает его к экземпляру класса CCanvas.

```
bool  Attach(
   const long          chart_id,                          // идентификатор чарта
   const string        objname,                           // имя объекта
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // способ обработки цвета
   )
```

Создает графический ресурс для объекта OBJ_BITMAP_LABEL и привязывает его к экземпляру класса CCanvas.

```
bool  Attach(
   const long          chart_id,                          // идентификатор чарта
   const string        objname,                           // имя объекта
   const int           width,                             // ширина картинки в пик
   const int           height,                            // высота картинки в пик
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // способ обработки цвета
   )
```

Параметры

chart_id

[out] Иидентификатор чарта.

objname

[in] Наименование (имя) графического объекта.

width

[in] Ширина картинки в ресурсе.

height

[in] Высота картинки в ресурсе.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Способ обработки альфа-канала. По умолчанию альфа-канал игнорируется.

Возвращаемое значение

true - в случае удачи, false - если не удалось привязать объект.

© 2000-2025, MetaQuotes Ltd.

Arc

Рисует дугу эллипса, вписанного в прямоугольник с углами в (x1,y1) и (x2,y2). Границы дуги отсекаются линиями из центра эллипса, идущими к двум точкам с координатами (x3,y3) и (x4,y4).

```
void  Arc(
   int          x1,      // координата X верхнего левого угла прямоугольника
   int          y1,      // координата Y верхнего левого угла прямоугольника
   int          x2,      // координата X нижнего правого угла прямоугольника
   int          y2,      // координата Y нижнего правого угла прямоугольника
   int          x3,      // координата X первой точки для нахождения границы дуги
   int          y3,      // координата Y первой точки для нахождения границы дуги
   int          x4,      // координата X второй точки для нахождения границы дуги
   int          y4,      // координата Y второй точки для нахождения границы дуги
   const uint   clr      // цвет
   );
```

Параметры

x1

[in] Координата X левого верхнего угла, определяющего прямоугольник.

y1

[in] Координата Y левого верхнего угла, определяющего прямоугольник.

x2

[in] Координата X правого нижнего угла, определяющего прямоугольник.

y2

[in] Координата Y правого нижнего угла, определяющего прямоугольник.

x3

[in] Координата X первой точки, к которой проведена линия из центра прямоугольника для получения границы дуги.

y3

[in] Координата Y первой точки, к которой проведена линия из центра прямоугольника для получения границы дуги.

x4

[in] Координата X второй точки, к которой проведена линия из центра прямоугольника для получения границы дуги.

y4

[in] Координата Y второй точки, к которой проведена линия из центра прямоугольника для получения границы дуги.

clr

[in] Цвет в формате ARGB. Для преобразования цвета в формат ARGB используйте функцию ColorToARGB().

© 2000-2025, MetaQuotes Ltd.

Рисует дугу эллипса с центром в точке (x,y), вписанного в прямоугольник с радиусами rx и ry. Границы дуги отсекаются лучами из центра эллипса, заданными углами fi3 и fi4.

```
void  Arc(
   int          x,        // координата X центра эллипса
   int          y,        // координата Y центра эллипса
   int          rx,       // радиус эллипса по координате X
   int          ry,       // радиус эллипса по координате Y
   int          fi3,      // угол луча из центра эллипса, задающий первую границу дуги
   int          fi4,      // угол луча из центра эллипса, задающий вторую границу дуги
   const uint   clr       // цвет
   );
```

Рисует дугу эллипса с центром в точке (x,y), вписанного в прямоугольник с радиусами rx и ry, и возвращает координаты границ дуги. Границы дуги отсекаются лучами из центра эллипса, заданными углами fi3 и fi4.

```
void  Arc(
   int          x,        // координата X центра эллипса
   int          y,        // координата Y центра эллипса
   int          rx,       // радиус эллипса по координате X
   int          ry,       // радиус эллипса по координате Y
   int          fi3,      // угол луча из центра эллипса, задающий первую границу дуги
   int          fi4,      // угол луча из центра эллипса, задающий вторую границу дуги
   int&         x3,       // координата X первой границы дуги
   int&         y3,       // координата Y первой границы дуги
   int&         x4,       // координата X второй границы дуги
   int&         y4,       // координата Y второй границы дуги
   const uint   clr       // цвет
   );
```

Параметры

x

[in] Координата X центра эллипса.

y

[in] Координата Y центра эллипса.

rx

[in] Радиус эллипса по координате X, в пикселях.

ry

[in] Радиус эллипса по координате Y, в пикселях.

fi3

[in] Угол в радианах, задающий первую границу дуги

fi4

[in] Угол в радианах, задающий вторую границу дуги

x3

© 2000-2025, MetaQuotes Ltd.

[out] Переменная для получения координаты X первой границы дуги.

y3

[out] Переменная для получения координаты Y первой границы дуги.

x4

[out] Переменная для получения координаты X второй границы дуги.

y4

[out] Переменная для получения координаты Y второй границы дуги.

clr

[in] Цвет в формате ARGB. Для преобразования цвета в формат ARGB используйте функцию ColorToARGB().

Примеры вызова методов класса:

```
#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,
   canvas.CircleAA(x1,y1,3, ColorToARGB(clrRed));
   canvas.CircleAA(x2,y2,3, ColorToARGB(clrBlue));
//--- show updated canvas
   canvas.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

Pie

Рисует закрашенный сектор эллипса, вписанного в прямоугольник с углами в (x1,y1) и (x2,y2). Границы сектора отсекаются линиями из центра эллипса, идущими к двум точкам с координатами (x3,y3) и (x4,y4).

```
void  Pie(
   int          x1,      // координата X верхнего левого угла прямоугольника
   int          y1,      // координата Y верхнего левого угла прямоугольника
   int          x2,      // координата X нижнего правого угла прямоугольника
   int          y2,      // координата Y нижнего правого угла прямоугольника
   int          x3,      // координата X первой точки для нахождения границы дуги
   int          y3,      // координата Y первой точки для нахождения границы дуги
   int          x4,      // координата X второй точки для нахождения границы дуги
   int          y4,      // координата Y второй точки для нахождения границы дуги
   const uint   clr,     // цвет линии
   const uint   fill_clr // цвет заливки
   );
```

Параметры

x1

[in] Координата X левого верхнего угла, определяющего прямоугольник.

y1

[in] Координата Y левого верхнего угла, определяющего прямоугольник.

x2

[in] Координата X правого нижнего угла, определяющего прямоугольник.

y2

[in] Координата Y правого нижнего угла, определяющего прямоугольник.

x3

[in] Координата X первой точки, к которой проведена линия из центра прямоугольника для получения границы дуги.

y3

[in] Координата Y первой точки, к которой проведена линия из центра прямоугольника для получения границы дуги.

x4

[in] Координата X второй точки, к которой проведена линия из центра прямоугольника для получения границы дуги.

y4

[in] Координата Y второй точки, к которой проведена линия из центра прямоугольника для получения границы дуги.

clr

[in] Цвет границы сектора в формате ARGB.

fill_clr

© 2000-2025, MetaQuotes Ltd.

[in] Цвет заливки сектора в формате ARGB. Для преобразования цвета в формат ARGB используйте функцию ColorToARGB().

Рисует закрашенный сектор эллипса с центром в точке (x,y), вписанного в прямоугольник с радиусами rx и ry. Границы сектора отсекаются лучами из центра эллипса, заданными углами fi3 и fi4.

```
void  Pie(
   int          x,        // координата X центра эллипса
   int          y,        // координата Y центра эллипса
   int          rx,       // радиус эллипса по координате X
   int          ry,       // радиус эллипса по координате Y
   int          fi3,      // угол луча из центра эллипса, задающий первую границу дуги
   int          fi4,      // угол луча из центра эллипса, задающий вторую границу дуги
   const uint   clr,      // цвет линии
   const uint   fill_clr // цвет заливки
   );
```

Рисует закрашенный сектор эллипса с центром в точке (x,y), вписанного в прямоугольник с радиусами rx и ry, и возвращает координаты границ дуги. Границы сектора отсекаются лучами из центра эллипса, заданными углами fi3 и fi4.

```
void  Pie(
   int          x,        // координата X центра эллипса
   int          y,        // координата Y центра эллипса
   int          rx,       // радиус эллипса по координате X
   int          ry,       // радиус эллипса по координате Y
   int          fi3,      // угол луча из центра эллипса, задающий первую границу дуги
   int          fi4,      // угол луча из центра эллипса, задающий вторую границу дуги
   int&         x3,       // координата X первой границы дуги
   int&         y3,       // координата Y первой границы дуги
   int&         x4,       // координата X второй границы дуги
   int&         y4,       // координата Y второй границы дуги
   const uint   clr,      // цвет линии
   const uint   fill_clr // цвет заливки
   );
```

Параметры

x

[in] Координата X центра эллипса.

y

[in] Координата Y центра эллипса.

rx

[in] Радиус эллипса по координате X, в пикселях.

ry

© 2000-2025, MetaQuotes Ltd.

[in] Радиус эллипса по координате X, в пикселях.

fi3

[in] Угол в радианах, задающий первую границу дуги

fi4

[in] Угол в радианах, задающий вторую границу дуги

x3

[out] Переменная для получения координаты X первой границы дуги.

y3

[out] Переменная для получения координаты Y первой границы дуги.

x4

[out] Переменная для получения координаты X второй границы дуги.

y4

[out] Переменная для получения координаты Y второй границы дуги.

clr

[in] Цвет границы сектора в формате ARGB.

fill_clr

[in] Цвет заливки сектора в формате ARGB. Для преобразования цвета в формат ARGB используйте функцию ColorToARGB().

Примеры вызова методов класса:

```
#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
```

© 2000-2025, MetaQuotes Ltd.

```
   canvas.Update();
   DebugBreak();
  }
```

© 2000-2025, MetaQuotes Ltd.

FillPolygon

Рисует закрашенный многоугольник.

```
void  FillPolygon(
   int&          x,       // массив с координатами X точек многоугольника
   int&          y,       // массив с координатами Y точек многоугольника
   const uint    clr      // цвет
   );
```

Параметры

x

[in] Массив, содержащий координаты X точек многоугольника.

y

[in] Массив, содержащий координаты Y точек многоугольника.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

FillEllipse

Рисует закрашенный эллипс, вписанный в прямоугольник с заданными координатами.

```
void  FillPolygon(
   int          x1,       // координата X верхнего левого угла прямоугольника
   int          y1,       // координата Y верхнего левого угла прямоугольника
   int          x2,       // координата X нижнего правого угла прямоугольника
   int          y2,       // координата Y нижнего правого угла прямоугольника
   const uint   clr        // цвет эллипса
   );
```

Параметры

x1

[in] Координата X левого верхнего угла, определяющего прямоугольник.

y1

[in] Координата Y левого верхнего угла, определяющего прямоугольник.

x2

[in] Координата X правого нижнего угла, определяющего прямоугольник.

y2

[in] Координата Y правого нижнего угла, определяющего прямоугольник.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

GetDefaultColor

Возвращает предопределенный цвет по его индексу.

```
static uint  GetDefaultColor(
   const uint  i      // индекс
   );
```

Параметры

i

[in] Индекс для получения цвета.

Возвращаемое значение

Цвет.

© 2000-2025, MetaQuotes Ltd.

ChartObjectName

Получает имя привязанного объекта чарта.

```
string  ChartObjectName();
```

Возвращаемое значение

имя привязанного объекта чарта

© 2000-2025, MetaQuotes Ltd.

Circle

Рисует окружность

```
void  Circle(
   int          x,        // координата X
   int          y,        // координата Y
   int          r,        // радиус
   const uint   clr       // цвет
   );
```

Параметры

x

[in] Координата X центра окружности.

y

[in] Координата Y центра окружности.

r

[in] Радиус окружности.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

CircleAA

Рисует окружность с использованием алгоритма сглаживания

```
void  CircleAA(
   const int      x,        // координата X
   const int      y,        // координата Y
   const double   r,        // радиус
   const uint     clr       // цвет
   );
```

Параметры

x

[in] Координата X центра окружности.

y

[in] Координата Y центра окружности.

r

[in] Радиус окружности.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

CircleWu

Рисует окружность с использованием алгоритма сглаживания By

```
void  CircleWu(
   const int     x,        // координата X
   const int     y,        // координата Y
   const double  r,        // радиус
   const uint    clr       // цвет
   );
```

Параметры

x

[in] Координата X центра окружности.

y

[in] Координата Y центра окружности.

r

[in] Радиус окружности.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический ресурс без привязки к объекту чарта.

```
virtual bool  Create(
   const string       name,                                  // имя
   const int          width,                                 // ширина
   const int          height,                                // высота
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // формат
   );
```

Параметры

name

[in] Основание для имени графического ресурса. Имя ресурса формируется при создании путем добавления псевдослучайной строки.

width

[in] Ширина (размер по оси X) в пикселях.

height

[in] Высота (размер по оси Y) в пикселях.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Способ обработки цвета. Более подробно о способах обработки цвета смотрите в описании функции ResourceCreate().

Возвращаемое значение

true - в случае удачи, иначе false

© 2000-2025, MetaQuotes Ltd.

CreateBitmap

Создает графический ресурс, привязанный к объекту чарта.

1. Создает графический ресурс в главном окне текущего чарта.

```
bool  CreateBitmap(
   const string       name,                                  // имя
   const datetime     time,                                  // время
   const double       price,                                 // цена
   const int          width,                                 // ширина
   const int          height,                                // высота
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // формат
   );
```

2. Создает графический ресурс с использованием идентификатора графика и номера подокна.

```
bool  CreateBitmap(
   const long         chart_id,                              // идентификатор чарта
   const int          subwin,                                // номер подокна
   const string       name,                                  // имя
   const datetime     time,                                  // время
   const double       price,                                 // цена
   const int          width,                                 // ширина
   const int          height,                                // высота
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // формат
   );
```

Параметры

chart_id

[in] Идентификатор чарта для создания объекта.

subwin

[in] Номер подокна чарта для создания объекта.

name

[in] Имя объекта чарта и основание для имени графического ресурса.

time

[in] Координата времени точки привязки объекта на чарте.

price

[in] Координата цены точки привязки объекта на чарте.

width

[in] Ширина графического ресурса (размер по оси X) в пикселях.

height

[in] Высота графического ресурса (размер по оси Y) в пикселях.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2025, MetaQuotes Ltd.

[in] Способ обработки цвета. Более подробно о способах обработки цвета смотрите в описании функции ResourceCreate().

Возвращаемое значение

true - в случае удачи, иначе - false

Примечание

Если используется первый вариант функции, то объект создается в главном окне текущего чарта.

Размеры объекта совпадают с размерами графического ресурса.

© 2000-2025, MetaQuotes Ltd.

CreateBitmapLabel

Создает графический ресурс, привязанный к объекту чарта.

1. Создает графический ресурс в главном окне текущего чарта.

```
bool  CreateBitmapLabel(
   const string       name,                                  // имя
   const int          x,                                     // координата X
   const int          y,                                     // координата Y
   const int          width,                                 // ширина
   const int          height,                                // высота
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // формат
   );
```

2. Создает графический ресурс с использованием идентификатора графика и номера подокна.

```
bool  CreateBitmapLabel(
   const long         chart_id,                              // идентификатор чарта
   const int          subwin,                                // номер подокна
   const string       name,                                  // имя
   const int          x,                                     // координата X
   const int          y,                                     // координата Y
   const int          width,                                 // ширина
   const int          height,                                // высота
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // формат
   );
```

Параметры

chart_id

[in] Идентификатор чарта для создания объекта.

subwin

[in] Номер подокна чарта для создания объекта.

name

[in] Имя объекта чарта и основание для имени графического ресурса.

x

[in] Координата X точки привязки объекта на чарте.

y

[in] Координата Y точки привязки объекта на чарте.

width

[in] Ширина графического ресурса (размер по оси X) в пикселях.

height

[in] Высота графического ресурса (размер по оси Y) в пикселях.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2025, MetaQuotes Ltd.

[in] Способ обработки цвета. Более подробно о способах обработки цвета смотрите в описании функции ResourceCreate().

Возвращаемое значение

true - в случае удачи, иначе false

Примечание

Если используется первый вариант функции, то объект создается в главном окне текущего чарта.

Размеры объекта совпадают с размерами графического ресурса.

© 2000-2025, MetaQuotes Ltd.

Destroy

Уничтожает графический ресурс.

```
void  Destroy();
```

Примечание

Если графический ресурс создавался с привязкой к объекту чарта, то объект чарта удаляется.

© 2000-2025, MetaQuotes Ltd.

Ellipse

Рисует эллипс по двум точкам.

```
void  Ellipse(
   int          x1,       // координата X
   int          y1,       // координата Y
   int          x2,       // координата X
   int          y2,       // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x1

[in] Координата X первой точки, определяющей эллипс.

y1

[in] Координата Y первой точки, определяющей эллипс.

x2

[in] Координата X второй точки, определяющей эллипс.

y2

[in] Координата Y второй точки, определяющей эллипс.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

EllipseAA

Рисует эллипс по двум точкам с использованием алгоритма сглаживания.

```
void  EllipseAA(
   int          x1,       // координата X
   int          y1,       // координата Y
   int          x2,       // координата X
   int          y2,       // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x1

[in] Координата X первой точки, определяющей эллипс.

y1

[in] Координата Y первой точки, определяющей эллипс.

x2

[in] Координата X второй точки, определяющей эллипс.

y2

[in] Координата Y второй точки, определяющей эллипс.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

EllipseWu

Рисует эллипс по двум точкам с использованием алгоритма сглаживания By.

```
void  Ellipse(
   int          x1,       // координата X
   int          y1,       // координата Y
   int          x2,       // координата X
   int          y2,       // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x1

[in] Координата X первой точки, определяющей эллипс.

y1

[in] Координата Y первой точки, определяющей эллипс.

x2

[in] Координата X второй точки, определяющей эллипс.

y2

[in] Координата Y второй точки, определяющей эллипс.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

Erase

Очищает или заполняет указанным цветом.

```
void  Erase(
   const uint  clr=0      // цвет
   );
```

Параметры

clr=0

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

Fill

Закрашивает область.

```
void  Fill(
   int          x,        // координата X
   int          y,        // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x

[in] Координата X точки начала закрашивания.

y

[in] Координата Y точки начала закрашивания.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

FillCircle

Рисует закрашенный круг.

```
void  FillCircle(
   int          x,        // координата X
   int          y,        // координата Y
   int          r,        // радиус
   const uint   clr       // цвет
   );
```

Параметры

x

[in] Координата X центра круга.

y

[in] Координата Y центра круга.

r

[in] Радиус круга.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

FillRectangle

Рисует закрашенный прямоугольник.

```
void  FillRectangle(
   int          x1,       // координата X
   int          y1,       // координата Y
   int          x2,       // координата X
   int          y2,       // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x1

[in] Координата X первой точки, определяющей прямоугольник.

y1

[in] Координата Y первой точки, определяющей прямоугольник.

x2

[in] Координата X второй точки, определяющей прямоугольник.

y2

[in] Координата Y второй точки, определяющей прямоугольник.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

FillTriangle

Рисует закрашенный треугольник.

```
void  FillTriangle(
   int          x1,       // координата X
   int          y1,       // координата Y
   int          x2,       // координата X
   int          y2,       // координата Y
   int          x3,       // координата X
   int          y3,       // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x1

[in] Координата X первой вершины треугольника.

y1

[in] Координата Y первой вершины треугольника.

x2

[in] Координата X второй вершины треугольника.

y2

[in] Координата Y второй вершины треугольника.

x3

[in] Координата X третьей вершины треугольника.

y3

[in] Координата Y третьей вершины треугольника.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

FontAngleGet

Получает угол наклона шрифта.

```
uint  FontAngleGet();
```

Возвращаемое значение

угол наклона шрифта

© 2000-2025, MetaQuotes Ltd.

FontAngleSet

Устанавливает угол наклона шрифта.

```
bool  FontAngleSet(
   uint  angle      // угол
   );
```

Параметры

angle

[in] Угол наклона шрифта в десятых долях градуса.

Возвращаемое значение

true - в случае удачи, иначе - false

© 2000-2025, MetaQuotes Ltd.

FontFlagsGet

Получает флаги шрифта.

```
uint  FontFlagsGet();
```

Возвращаемое значение

флаги шрифта

© 2000-2025, MetaQuotes Ltd.

FontFlagsSet

Устанавливает флаги шрифта.

```
bool  FontFlagsSet(
   uint  flags      // флаги
   );
```

Параметры

flags

[in] Флаги создания шрифта. Более подробно о флагах смотрите в описании функции TextSetFont().

Возвращаемое значение

true - в случае удачи, иначе - false

© 2000-2025, MetaQuotes Ltd.

FontGet

Получает параметры текущего шрифта.

```
void  FontGet(
   string&  name,       // имя
   int&     size,       // размер
   uint&    flags,      // флаги
   uint&    angle       // угол наклона
   );
```

Параметры

name

[out] Ссылка на переменную для возврата имени шрифта.

size

[out] Ссылка на переменную для возврата размера шрифта.

flags

[out] Ссылка на переменную для возврата флагов шрифта.

angle

[out] Ссылка на переменную для возврата угла наклона шрифта.

© 2000-2025, MetaQuotes Ltd.

FontNameGet

Получает имя шрифта.

```
string  FontNameGet();
```

Возвращаемое значение

имя шрифта

© 2000-2025, MetaQuotes Ltd.

FontNameSet

Устанавливает имя шрифта.

```
bool  FontNameSet(
   string  name      // имя
   );
```

Параметры

name

[in] Имя шрифта. Например, "Arial".

Возвращаемое значение

true - в случае удачи, иначе - false

© 2000-2025, MetaQuotes Ltd.

FontSet

Устанавливает текущий шрифт.

```
bool  FontSet(
   const string  name,         // имя
   const int     size,         // размер
   const uint    flags=0,      // флаги
   const uint    angle=0       // угол
   );
```

Параметры

name

[in] Имя шрифта. Например "Arial".

size

[in] Размер шрифта. Более подробно о специфике задания размера смотрите в описании функции TextSetFont().

flags=0

[in] Флаги создания шрифта. Более подробно о флагах смотрите в описании функции TextSetFont().

angle=0

[in] Угол наклона шрифта в десятых долях градуса.

Возвращаемое значение

true - в случае удачи, иначе - false

© 2000-2025, MetaQuotes Ltd.

FontSizeGet

Получает размер шрифта.

```
int  FontSizeGet();
```

Возвращаемое значение

размер шрифта

© 2000-2025, MetaQuotes Ltd.

FontSizeSet

Устанавливает размер шрифта.

```
bool  FontSizeSet(
   int  size      // размер
   );
```

Параметры

size

[in] Размер шрифта. Более подробно о специфике задания размера смотрите в описании функции TextSetFont().

Возвращаемое значение

true - в случае удачи, иначе - false

© 2000-2025, MetaQuotes Ltd.

Height

Получает высоту графического ресурса.

```
int  Height();
```

Возвращаемое значение

высота графического ресурса

© 2000-2025, MetaQuotes Ltd.

Line

Рисует отрезок произвольной линии.

```
void  Line(
   int          x1,       // координата X
   int          y1,       // координата Y
   int          x2,       // координата X
   int          y2,       // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x1

[in] Координата X первой точки отрезка.

y1

[in] Координата Y первой точки отрезка.

x2

[in] Координата X второй точки отрезка.

y2

[in] Координата Y второй точки отрезка.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

LineAA

Рисует отрезок произвольной линии с использованием алгоритма сглаживания.

```
void  LineAA(
   const int   x1,                  // координата X
   const int   y1,                  // координата Y
   const int   x2,                  // координата X
   const int   y2,                  // координата Y
   const uint  clr,                 // цвет
   const uint  style=UINT_MAX       // стиль линии
   );
```

Параметры

x1

[in] Координата X первой точки отрезка.

y1

[in] Координата Y первой точки отрезка.

x2

[in] Координата X второй точки отрезка.

y2

[in] Координата Y второй точки отрезка.

clr

[in] Цвет в формате ARGB.

style=UINT_MAX

[in] Стиль линии - одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

© 2000-2025, MetaQuotes Ltd.

LineWu

Рисует отрезок произвольной линии с использованием алгоритма сглаживания By.

```
void  LineWu(
   const int   x1,                  // координата X
   const int   y1,                  // координата Y
   const int   x2,                  // координата X
   const int   y2,                  // координата Y
   const uint  clr,                 // цвет
   const uint  style=UINT_MAX       // стиль линии
   );
```

Параметры

x1

[in] Координата X первой точки отрезка.

y1

[in] Координата Y первой точки отрезка.

x2

[in] Координата X второй точки отрезка.

y2

[in] Координата Y второй точки отрезка.

clr

[in] Цвет в формате ARGB.

style=UINT_MAX

[in] Стиль линии - одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

© 2000-2025, MetaQuotes Ltd.

LineHorizontal

Рисует отрезок горизонтальной линии.

```
void  LineHorizontal(
   int          x1,       // координата X
   int          x2,       // координата X
   int          y,        // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x1

[in] Координата X первой точки отрезка.

x2

[in] Координата X второй точки отрезка.

y

[in] Координата Y отрезка.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

LineVertical

Рисует отрезок вертикальной линии.

```
void  LineVertical(
   int          x,        // координата X
   int          y1,       // координата Y
   int          y2,       // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x

[in] Координата X отрезка.

y1

[in] Координата Y первой точки отрезка.

y2

[in] Координата Y второй точки отрезка.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

LineStyleSet

Устанавливает стиль линии.

```
void  LineStyleSet(
   const uint  style      // стиль
   );
```

Параметры

style

[in] Стиль линии.

Примечание

Входной параметр может принимать любое из значений перечисления ENUM_LINE_STYLE. Кроме того, существует возможность создать пользовательский стиль рисования линии.

© 2000-2025, MetaQuotes Ltd.

LineThick

Рисует отрезок произвольной линии заданной толщины с использованием алгоритма сглаживания.

```
void  LineThick(
   const int       x1,            // координата X первой точки отрезка
   const int       y1,            // координата Y первой точки отрезка
   const int       x2,            // координата X второй точки отрезка
   const int       y2,            // координата Y второй точки отрезка
   const uint      clr,           // цвет
   const int       size,          // толщина линии
   const uint      style,         // стиль линии
   ENUM_LINE_END   end_style      // стиль концов линии
   )
```

Параметры

x1

[in] Координата X первой точки отрезка.

y1

[in] Координата Y первой точки отрезка.

x2

[in] Координата X второй точки отрезка.

y2

[in] Координата Y второй точки отрезка.

clr

[in] Цвет в формате ARGB.

size

[in] Толщина линии.

style

[in] Стиль линии — одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

end_style

[in] Стиль концов линии — одно из значений перечисления ENUM_LINE_END

ENUM_LINE_END

Идентификатор	Описание
LINE_END_ROUND	Концы линий закругляются.
LINE_END_BUTT	Концы линий обрезаются.

© 2000-2025, MetaQuotes Ltd.

Идентификатор	Описание
LINE_END_SQUARE	Линия оканчивается закрашенным прямоугольником.

© 2000-2025, MetaQuotes Ltd.

LineThickVertical

Рисует вертикальный отрезок произвольной линии заданной толщины с использованием алгоритма сглаживания.

```
void  LineThickVertical(
   const int       x,              // координата X отрезка
   const int       y1,             // координата Y первой точки отрезка
   const int       y2,             // координата Y второй точки отрезка
   const uint      clr,            // цвет
   const int       size,           // толщина линии
   const uint      style,          // стиль линии
   ENUM_LINE_END   end_style       // стиль концов линии
   )
```

Параметры

x

[in] Координата X отрезка.

y1

[in] Координата Y первой точки отрезка.

y2

[in] Координата Y второй точки отрезка.

clr

[in] Цвет в формате ARGB.

size

[in] Толщина линии.

style

[in] Стиль линии — одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

end_style

[in] Стиль концов линии — одно из значений перечисления ENUM_LINE_END.

© 2000-2025, MetaQuotes Ltd.

LineThickHorizontal

Рисует горизонтальный отрезок произвольной линии заданной толщины со сглаживанием.

```
void  LineThickHorizontal(
   const int       x1,            // координата X первой точки отрезка
   const int       x2,            // координата X второй точки отрезка
   const int       y,             // координата Y отрезка
   const uint      clr,           // цвет
   const int       size,          // толщина линии
   const uint      style,         // стиль линии
   ENUM_LINE_END   end_style      // стиль концов линии
   )
```

Параметры

x1

[in] Координата X первой точки отрезка.

x2

[in] Координата X второй точки отрезка.

y

[in] Координата Y отрезка.

clr

[in] Цвет в формате ARGB.

size

[in] Толщина линии.

style

[in] Стиль линии — одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

end_style

[in] Стиль концов линии — одно из значений перечисления ENUM_LINE_END.

© 2000-2025, MetaQuotes Ltd.

LoadFromFile

Читает рисунок из BMP-файла.

```
bool  LoadFromFile(
   const string  filename      // имя файла
   );
```

Параметры

filename

[in] Имя файла (включая расширение "BMP").

Возвращаемое значение

true - в случае удачи, иначе - false

© 2000-2025, MetaQuotes Ltd.

PixelGet

Получает цвет точки с указанными координатами.

```
uint  PixelGet(
   const int  x,      // координата X
   const int  y       // координата Y
   );
```

Параметры

x

[in] Координата X точки.

y

[in] Координата Y точки.

Возвращаемое значение

цвет точки в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

PixelSet

Устанавливает цвет точки с указанными координатами.

```
void  PixelSet(
   const int   x,        // координата X
   const int   y,        // координата Y
   const uint  clr       // цвет
   );
```

Параметры

x

[in] Координата X точки.

y

[in] Координата Y точки.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

PixelSetAA

Рисует точку с использованием алгоритма сглаживания.

```
void  PixelSetAA(
   const double  x,        // координата X
   const double  y,        // координата Y
   const uint    clr       // цвет
   );
```

Параметры

x

[in] Координата X точки.

y

[in] Координата Y точки.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

Polygon

Рисует многоугольник.

```
void  Polygon(
   int&         x[],     // массив координат X
   int&         y[],     // массив координат Y
   const uint   clr      // цвет
   );
```

Параметры

x[]

[in] Массив координат X точек многоугольника.

y[]

[in] Массив координат Y точек многоугольника.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

PolygonAA

Рисует многоугольник с использованием алгоритма сглаживания.

```
void  PolygonAA(
   int&         x[],                // массив координат X
   int&         y[],                // массив координат Y
   const uint   clr,                // цвет
   const uint   style=UINT_MAX      // стиль линии
   );
```

Параметры

x[]

[in] Массив координат X точек многоугольника.

y[]

[in] Массив координат Y точек многоугольника.

clr

[in] Цвет в формате ARGB.

style=UINT_MAX

[in] Стиль линии - одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

© 2000-2025, MetaQuotes Ltd.

PolygonWu

Рисует многоугольник с использованием алгоритма сглаживания By.

```
void  PolygonWu(
   int&         x[],                // массив координат X
   int&         y[],                // массив координат Y
   const uint   clr,                // цвет
   const uint   style=UINT_MAX      // стиль линии
   );
```

Параметры

x[]

[in] Массив координат X точек многоугольника.

y[]

[in] Массив координат Y точек многоугольника.

clr

[in] Цвет в формате ARGB.

style=UINT_MAX

[in] Стиль линии - одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

© 2000-2025, MetaQuotes Ltd.

PolygonThick

Рисует многоугольник заданной толщины с использованием алгоритма сглаживания.

```
void  PolygonThick(
   const int&      x[],          // массив координат X точек многоугольника
   const int&      y[],          // массив координат Y точек многоугольника
   const uint      clr,          // цвет
   const int       size,         // толщина линии
   const uint      style,        // стиль линии
   ENUM_LINE_END   end_style     // стиль концов линии
   )
```

Параметры

x[]

[in] Массив координат X точек многоугольника.

y[]

[in] Массив координат Y точек многоугольника.

clr

[in] Цвет в формате ARGB.

size

[in] Толщина линии.

style

[in] Стиль линии — одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

end_style

[in] Стиль концов линии — одно из значений перечисления ENUM_LINE_END.

© 2000-2025, MetaQuotes Ltd.

PolygonSmooth

Рисует многоугольник заданной толщины с использованием двух алгоритмов сглаживания последовательно. Сначала на основе кривых Безье сглаживаются отдельные отрезки. Затем для повышения качества отрисовки к построенному из этих отрезков многоугольнику применяется растровый алгоритм сглаживания.

```
void  PolygonSmooth(
   int&              x[],                        // массив координат X точек многоуго
   int&              y[],                        // массив координат Y точек многоуго
   const uint        clr,                        // цвет
   const int         size,                       // толщина линии
   ENUM_LINE_STYLE   style=STYLE_SOLID,          // стиль линии
   ENUM_LINE_END     end_style=LINE_END_ROUND,   // стиль концов линии
   double            tension=0.5,                // значение параметра сглаживания
   double            step=10                     // длина аппроксимирующих линий
   )
```

Параметры

&x[]

[in] Массив координат X точек многоугольника.

&y[]

[in] Массив координат Y точек многоугольника.

clr

[in] Цвет в формате ARGB.

size

[in] Толщина линии.

style=STYLE_SOLID

[in] Стиль линии — одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

end_style=LINE_END_ROUND

[in] Стиль концов линии — одно из значений перечисления ENUM_LINE_END.

tension=0.5

[in] Значение параметра сглаживания.

step=10

[in] Длина аппроксимирующих линий.

© 2000-2025, MetaQuotes Ltd.

Polyline

Рисует ломаную линию.

```
void  Polyline(
   int&         x[],      // массив координат X
   int&         y[],      // массив координат Y
   const uint   clr       // цвет
   );
```

Параметры

x[]

[in] Массив координат X точек ломаной линии.

y[]

[in] Массив координат Y точек ломаной линии.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

PolylineSmooth

Рисует ломаную линию заданной толщины с использованием двух алгоритмов сглаживания последовательно. Сначала на основе кривых Безье сглаживаются отдельные отрезки линии. Затем для повышения качества отрисовки к построенной из этих отрезков ломаной линии применяется растровый алгоритм сглаживания.

```
void  PolylineSmooth(
   const int&        x[],                        // массив координат X точек ломаной л
   const int&        y[],                        // массив координат Y точек ломаной л
   const uint        clr,                        // цвет
   const int         size,                       // толщина линии
   ENUM_LINE_STYLE   style=STYLE_SOLID,          // стиль линии
   ENUM_LINE_END     end_style=LINE_END_ROUND,   // стиль концов линии
   double            tension=0.5,                // значение параметра сглаживания
   double            step=10                     // шаг аппроксимации
   )
```

Параметры

&x[]

[in] Массив координат X точек ломаной линии.

&y[]

[in] Массив координат Y точек ломаной линии.

clr

[in] Цвет в формате ARGB.

size

[in] Толщина линии.

style=STYLE_SOLID

[in] Стиль линии — одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

end_style=LINE_END_ROUND

[in] Стиль концов линии — одно из значений перечисления ENUM_LINE_END.

tension=0.5

[in] Значение параметра сглаживания.

step=10

[in] Шаг аппроксимации.

© 2000-2025, MetaQuotes Ltd.

PolylineThick

Рисует ломаную линию заданной толщины с использованием алгоритма сглаживания.

```
void  PolylineThick(
   const int       &x[],          // массив координат X точек ломаной линии
   const int       &y[],          // массив координат Y точек ломаной линии
   const uint      clr,           // цвет
   const int       size,          // толщина линии
   const uint      style,         // стиль линии
   ENUM_LINE_END   end_style      // стиль концов линии
   )
```

Параметры

&x[]

[in] Массив координат X точек ломаной линии.

&y[]

[in] Массив координат Y точек ломаной линии.

clr

[in] Цвет в формате ARGB.

size

[in] Толщина линии.

style

[in] Стиль линии — одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

end_style

[in] Стиль концов линии — одно из значений перечисления ENUM_LINE_END

© 2000-2025, MetaQuotes Ltd.

PolylineWu

Рисует ломаную линию с использованием алгоритма сглаживания By.

```
void  PolylineWu(
   int&         x[],                // массив координат X
   int&         y[],                // массив координат Y
   const uint   clr,                // цвет
   const uint   style=UINT_MAX      // стиль линии
   );
```

Параметры

x[]

[in] Массив координат X точек ломаной линии.

y[]

[in] Массив координат Y точек ломаной линии.

clr

[in] Цвет в формате ARGB.

style=UINT_MAX

[in] Стиль линии - одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

© 2000-2025, MetaQuotes Ltd.

PolylineAA

Рисует ломаную линию с использованием алгоритма сглаживания.

```
void  PolylineAA(
   int&         x[],                // массив координат X
   int&         y[],                // массив координат Y
   const uint   clr,                // цвет
   const uint   style=UINT_MAX      // стиль линии
   );
```

Параметры

x[]

[in] Массив координат X точек ломаной линии.

y[]

[in] Массив координат Y точек ломаной линии.

clr

[in] Цвет в формате ARGB.

style=UINT_MAX

[in] Стиль линии - одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

© 2000-2025, MetaQuotes Ltd.

Rectangle

Рисует прямоугольник по двум точкам.

```
void  Rectangle(
   int          x1,       // координата X
   int          y1,       // координата Y
   int          x2,       // координата X
   int          y2,       // координата Y
   const uint   clr       // цвет
   );
```

Параметры

x1

[in] Координата X первой точки, определяющей прямоугольник.

y1

[in] Координата Y первой точки, определяющей прямоугольник.

x2

[in] Координата X второй точки, определяющей прямоугольник.

y2

[in] Координата Y второй точки, определяющей прямоугольник.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

Resize

Изменяет размеры графического ресурса.

```
bool  Resize(
   const int  width,       // ширина
   const int  height       // высота
   );
```

Параметры

width

[in] Новая ширина графического ресурса.

height

[in] Новая высота графического ресурса.

Возвращаемое значение

true - в случае удачи, иначе - false

Примечание

При изменении размеров предыдущее изображение не сохраняется.

© 2000-2025, MetaQuotes Ltd.

ResourceName

Получает имя графического ресурса.

```
string  ResourceName();
```

Возвращаемое значение

имя графического ресурса

© 2000-2025, MetaQuotes Ltd.

TextHeight

Получает высоту текста.

```
int  TextHeight(
   const string  text      // текст
   );
```

Параметры

text

[in] Текст для измерения.

Возвращаемое значение

высота текста в пикселях

Примечание

Для измерения текста используется текущий шрифт.

© 2000-2025, MetaQuotes Ltd.

TextOut

Выводит текст.

```
void  TextOut(
   int          x,               // координата X
   int          y,               // координата Y
   string       text,            // текст
   const uint   clr,             // цвет
   uint         alignment=0      // выравнивание
   );
```

Параметры

x

[in] Координата X точки привязки текста.

y

[in] Координата Y точки привязки текста.

text

[in] Текст для отображения.

clr

[in] Цвет в формате ARGB.

alignment=0

[in] Способ привязки текста. Более подробно о способах привязки смотрите в описании функции работы с графическими объектами TextOut().

Примечание

Для вывода текста используется текущий шрифт.

© 2000-2025, MetaQuotes Ltd.

TextSize

Получает размеры текста.

```
void  TextSize(
   const string  text,        // текст
   int&          width,       // ширина
   int&          height       // высота
   );
```

Параметры

text

[in] Текст для измерения.

width

[out] Ссылка на переменную для возврата ширины текста.

height

[out] Ссылка на переменную для возврата высоты текста.

Примечание

Для измерения текста используется текущий шрифт.

© 2000-2025, MetaQuotes Ltd.

TextWidth

Получает ширину текста.

```
int  TextWidth(
   const string  text      // текст
   );
```

Параметры

text

[in] Текст для измерения.

Возвращаемое значение

ширина текста в пикселях

Примечание

Для измерения текста используется текущий шрифт.

© 2000-2025, MetaQuotes Ltd.

TransparentLevelSet

Устанавливает уровень прозрачности.

```
void  TransparentLevelSet(
   const uchar  value      // значение
   );
```

Параметры

value

[in] Новое значение уровня прозрачности.

Примечание

Значение 0 соответствует полной прозрачности, значение 255 - абсолютной непрозрачности.

Установка уровня прозрачности влияет на все ранее нарисованное. На последующие построения установленный уровень прозрачности не распространяется.

© 2000-2025, MetaQuotes Ltd.

Triangle

Рисует треугольник.

```
void  Triangle(
   int          x1,      // координата X
   int          y1,      // координата Y
   int          x2,      // координата X
   int          y2,      // координата Y
   int          x3,      // координата X
   int          y3,      // координата Y
   const uint   clr      // цвет
   );
```

Параметры

x1

[in] Координата X первой вершины треугольника.

y1

[in] Координата Y первой вершины треугольника.

x2

[in] Координата X второй вершины треугольника.

y2

[in] Координата Y второй вершины треугольника.

x3

[in] Координата X третьей вершины треугольника.

y3

[in] Координата Y третьей вершины треугольника.

clr

[in] Цвет в формате ARGB.

© 2000-2025, MetaQuotes Ltd.

TriangleAA

Рисует треугольник с использованием алгоритма сглаживания.

```
void  TriangleAA(
   const int   x1,                  // координата X
   const int   y1,                  // координата Y
   const int   x2,                  // координата X
   const int   y2,                  // координата Y
   const int   x3,                  // координата X
   const int   y3,                  // координата Y
   const uint  clr,                 // цвет
   const uint  style=UINT_MAX       // стиль линии
   );
```

Параметры

x1

[in] Координата X первой вершины треугольника.

y1

[in] Координата Y первой вершины треугольника.

x2

[in] Координата X второй вершины треугольника.

y2

[in] Координата Y второй вершины треугольника.

x3

[in] Координата X третьей вершины треугольника.

y3

[in] Координата Y третьей вершины треугольника.

clr

[in] Цвет в формате ARGB.

style=UINT_MAX

[in] Стиль линии - одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

© 2000-2025, MetaQuotes Ltd.

TriangleWu

Рисует треугольник с использованием алгоритма сглаживания By.

```
void  TriangleWu(
   const int   x1,                  // координата X
   const int   y1,                  // координата Y
   const int   x2,                  // координата X
   const int   y2,                  // координата Y
   const int   x3,                  // координата X
   const int   y3,                  // координата Y
   const uint  clr,                 // цвет
   const uint  style=UINT_MAX       // стиль линии
   );
```

Параметры

x1

[in] Координата X первой вершины треугольника.

y1

[in] Координата Y первой вершины треугольника.

x2

[in] Координата X второй вершины треугольника.

y2

[in] Координата Y второй вершины треугольника.

x3

[in] Координата X третьей вершины треугольника.

y3

[in] Координата Y третьей вершины треугольника.

clr

[in] Цвет в формате ARGB.

style=UINT_MAX

[in] Стиль линии - одно из значений перечисления ENUM_LINE_STYLE или пользовательское значение.

© 2000-2025, MetaQuotes Ltd.

Update

Отображает изменения на экран.

```
void  Update(
   const bool  redraw=true      // флаг
   );
```

Параметры

redraw=true

Флаг необходимости перерисовки чарта.

© 2000-2025, MetaQuotes Ltd.

Width

Получает ширину графического ресурса.

```
int  Width();
```

Возвращаемое значение

ширина графического ресурса

© 2000-2025, MetaQuotes Ltd.

CChartCanvas

Базовый класс для реализации классов, с помощью которых происходит отрисовка графиков и их элементов.

Описание

Данный класс включает в себя методы для работы с основными элементами любого графика: координатными осями и их разметкой, легендой графика, сеткой, фоном и т.д. Здесь настраиваются параметры отображения элементов: видимость/невидимость, цвет текста и т.д.

Декларация

```
class CChartCanvas : public CCanvas
```

Заголовок

```
#include <Canvas\Charts\ChartCanvas.mqh>
```

Иерархия наследования

CCanvas

CChartCanvas

Прямые потомки

CHistogramChart, CLineChart, CPieChart

Методы класса

Метод	Действие
ColorBackground	Возвращает и устанавливает цвет фона.
ColorBorder	Возвращает и устанавливает цвет границ.
ColorText	Возвращает и устанавливает цвет текста.
ColorGrid	Возвращает и устанавливает цвет сетки.
MaxData	Возвращает и устанавливает максимальное количество допустимых данных (серий).
MaxDescrLen	Возвращает и устанавливает максимальную длину дескрипторов.
ShowFlags	Возвращает и устанавливает флаг видимости элементов графика.
IsShowLegend	Возвращает флаг видимости легенды на графике.
IsShowScaleLeft	Возвращает флаг видимости шкалы значений слева.

© 2000-2025, MetaQuotes Ltd.

Метод	Действие
IsShowScaleRight	Возвращает флаг видимости шкалы значений справа.
IsShowScaleTop	Возвращает флаг видимости шкалы значений сверху.
IsShowScaleBottom	Возвращает флаг видимости шкалы значений снизу.
IsShowGrid	Возвращает флаг видимости сетки на графике.
IsShowDescriptors	Возвращает флаг видимости дескрипторов на графике.
IsShowPercent	Возвращает флаг видимости процентов на графике.
VScaleMin	Возвращает и устанавливает минимум на вертикальной шкале значений.
VScaleMax	Возвращает и устанавливает максимум на вертикальной шкале значений.
NumGrid	Возвращает и устанавливает количество вертикальных делений шкалы при построении сетки графика.
DataOffset	Возвращает и устанавливает значение смещения данных.
DataTotal	Возвращает общее количество серий данных на графике.
DrawDescriptors	Виртуальный метод для отрисовки дескрипторов.
DrawData	Виртуальный метод для отрисовки серии данных по указанному индексу.
Create	Виртуальный метод, который создаёт графический ресурс.
AllowedShowFlags	Устанавливает набор разрешенных флагов видимости для элементов графика.
ShowLegend	Устанавливает флаг видимости для легенды.
ShowScaleLeft	Устанавливает флаг видимости для левой шкалы.

© 2000-2025, MetaQuotes Ltd.

Метод	Действие
ShowScaleRight	Устанавливает флаг видимости для правой шкалы.
ShowScaleTop	Устанавливает флаг видимости для верхней шкалы.
ShowScaleBottom	Устанавливает флаг видимости для нижней шкалы.
ShowGrid	Устанавливает флаг видимости для сетки.
ShowDescriptors	Устанавливает флаг видимости для дескрипторов.
ShowValue	Устанавливает флаг видимости для значений.
ShowPercent	Устанавливает флаг видимости для процентов.
LegendAlignment	Устанавливает выравнивание текста для легенды.
Accumulative	Устанавливает флаг накопления значений для серий.
VScaleParams	Устанавливает параметры для вертикальной шкалы значений.
DescriptorUpdate	Обновляет значение дескриптора серии (по заданной позиции).
ColorUpdate	Обновляет цвета серий (по заданной позиции).
ValuesCheck	Производит внутренние вычисления для построения графика.
Redraw	Перерисовывает график.
DrawBackground	Отрисовывает фон.
DrawLegend	Перерисовывает легенду.
DrawLegendVertical	Отрисовывает вертикальную легенду.
DrawLegendHorizontal	Отрисовывает горизонтальную легенду.
CalcScales	Рассчитывает координаты шкалы.
DrawScales	Перерисовывает все шкалы значений.
DrawScaleLeft	Перерисовывает левую шкалу значений.
DrawScaleRight	Перерисовывает правую шкалу значений.
DrawScaleTop	Перерисовывает верхнюю шкалу значений

© 2000-2025, MetaQuotes Ltd.

Метод	Действие
DrawScaleBottom	Перерисовывает нижнюю шкалу значений.
DrawGrid	Перерисовывает сетку.
DrawChart	Перерисовывает данные.

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

ColorBackground (метод Get)

Возвращает цвет фона.

```
uint  ColorBackground()
```

Возвращаемое значение

Цвет фона.

ColorBackground (метод Set)

Устанавливает цвет заднего фона.

```
void  ColorBackground(
  const uint  value,  // цвет фона
  )
```

Параметры

value

[in] Цвет фона.

© 2000-2025, MetaQuotes Ltd.

ColorBorder (метод Get)

Возвращает цвет границ.

```
uint  ColorBorder()
```

Возвращаемое значение

Цвет границ.

ColorBorder (метод Set)

Устанавливает цвет границ.

```
void  ColorBorder(
  const uint  value,  // цвет границ
  )
```

Параметры

value

[in] Цвет границ.

© 2000-2025, MetaQuotes Ltd.

ColorText (метод Get)

Возвращает цвет текста.

```
uint  ColorText()
```

Возвращаемое значение

Цвет текста.

ColorText (метод Set)

Устанавливает цвет текста.

```
void  ColorText(
  const uint  value,  // цвет текста
  )
```

Параметры

value

[in] Цвет текста.

© 2000-2025, MetaQuotes Ltd.

ColorGrid (метод Get)

Возвращает цвет сетки.

```
uint  ColorGrid()
```

Возвращаемое значение

Цвет сетки.

ColorGrid (метод Set)

Устанавливает цвет сетки.

```
void  ColorGrid(
  const uint  value,  // цвет сетки
  )
```

Параметры

value

[in] Цвет сетки.

© 2000-2025, MetaQuotes Ltd.

MaxData (метод Get)

Возвращает максимальное количество допустимых данных (серий).

```
uint  MaxData()
```

Возвращаемое значение

Максимальное количество данных (серий).

MaxData (метод Set)

Устанавливает максимальное количество допустимых данных (серий).

```
void  MaxData(
  const uint  value,  // количество данных
  )
```

Параметры

value

[in] Максимальное количество данных (серий).

© 2000-2025, MetaQuotes Ltd.

MaxDescrLen (метод Get)

Возвращает максимальную длину дескрипторов.

```
uint  MaxDescrLen()
```

Возвращаемое значение

Значение максимальной длины дескрипторов.

MaxDescrLen (метод Set)

Устанавливает максимальную длину дескрипторов.

```
void  MaxDescrLen(
  const uint  value,  // максимальная длина
  )
```

Параметры

value

[in] Значение максимальной длины дескрипторов.

© 2000-2025, MetaQuotes Ltd.

ShowFlags (метод Get)

Возвращает флаг видимости элементов графика.

```
bool  ShowFlags()
```

Возвращаемое значение

Значение флага видимости элементов графика.

ShowFlags (метод Set)

Устанавливает флаг видимости элементов графика.

```
void  ShowFlags(
  const uint  flags,  // флаг
  )
```

Parameters

flags

[in] Значение флага видимости элементов графика.

© 2000-2025, MetaQuotes Ltd.

IsShowLegend

Возвращает флаг видимости легенды на графике.

```
bool  IsShowLegend()
```

Возвращаемое значение

true, если легенда видна, иначе false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleLeft

Возвращает флаг видимости шкалы значений слева.

```
bool  IsShowScaleLeft()
```

Возвращаемое значение

true, если шкала значений видна, иначе — false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleRight

Возвращает флаг видимости шкалы значений справа.

```
bool  IsShowScaleRight()
```

Возвращаемое значение

true, если шкала значений видна, иначе — false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleTop

Возвращает флаг видимости шкалы значений сверху.

```
bool  IsShowScaleTop()
```

Возвращаемое значение

true, если шкала значений видна, иначе — false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleBottom

Возвращает флаг видимости шкалы значений снизу.

```
bool  IsShowScaleBottom()
```

Возвращаемое значение

true, если шкала значений видна, иначе — false.

© 2000-2025, MetaQuotes Ltd.

IsShowGrid

Возвращает флаг видимости сетки на графике.

```
bool  IsShowGrid()
```

Возвращаемое значение

true, если сетка видна, иначе — false.

© 2000-2025, MetaQuotes Ltd.

IsShowDescriptors

Возвращает флаг видимости дескрипторов на графике.

```
bool  IsShowDescriptors()
```

Возвращаемое значение

true, если дескрипторы видны, иначе — false.

© 2000-2025, MetaQuotes Ltd.

IsShowPercent

Возвращает флаг видимости процентов на графике.

```
bool  IsShowPercent()
```

Возвращаемое значение

true, если проценты видны, иначе — false.

© 2000-2025, MetaQuotes Ltd.

VScaleMin (Метод Get)

Возвращает минимум на вертикальной шкале значений.

```
double  VScaleMin()
```

Возвращаемое значение

Минимальное значение на вертикальной шкале.

VScaleMin (Метод Set)

Устанавливает минимум на вертикальной шкале значений.

```
void  VScaleMin(
  const double  value,    // значение на вертикальной шкале
  )
```

Параметры

value

[in] Минимальное значение.

© 2000-2025, MetaQuotes Ltd.

VScaleMax

Возвращает максимум на вертикальной шкале значений.

```
double  VScaleMax()
```

Возвращаемое значение

Максимальное значение на вертикальной шкале.

VScaleMax

Устанавливает максимум на вертикальной шкале значений.

```
void  VScaleMax(
  const double  value,  //   значение на вертикальной шкале
  )
```

Параметры

value

[in] Максимальное значение.

© 2000-2025, MetaQuotes Ltd.

NumGrid

Возвращает количество вертикальных делений шкалы при построении сетки графика.

```
uint  NumGrid()
```

Возвращаемое значение

Количество делений.

NumGrid

Устанавливает количество вертикальных делений шкалы при построении сетки графика.

```
void  NumGrid(
  const uint  value,  // количество делений
  )
```

Параметры

value

[in] Количество делений.

© 2000-2025, MetaQuotes Ltd.

DataOffset

Возвращает значение смещения данных.

```
int  DataOffset()
```

Возвращаемое значение

Смещение данных.

DataOffset

Устанавливает значение смещения данных.

```
void  DataOffset(
  const int  value,  // значение
  )
```

Параметры

value

[in] Смещение данных.

© 2000-2025, MetaQuotes Ltd.

DataTotal

Возвращает общее количество серий данных на графике.

```
uint  DataTotal()
```

Возвращаемое значение

Количество серий.

© 2000-2025, MetaQuotes Ltd.

DrawDescriptors

Виртуальный метод для отрисовки дескрипторов.

```
virtual void  DrawDescriptors()
```

© 2000-2025, MetaQuotes Ltd.

DrawData

Виртуальный метод для отрисовки серии данных по указанному индексу.

```
virtual void  DrawData(
   const uint  idx=0,  // индекс
   )
```

Параметры

idx=0

[in] Индекс серии.

© 2000-2025, MetaQuotes Ltd.

Create

Виртуальный метод, который создаёт графический ресурс.

```
virtual bool  Create(
  const string       name,      // имя ресурса
  const int          width,     // ширина
  const int          height,    // высота
  ENUM_COLOR_FORMAT  clrfmt,    // формат
  )
```

Параметры

name

[in] Основание для имени графического ресурса. Имя ресурса формируется при создании путем добавления псевдослучайной строки.

width

[in] Ширина (размер по оси X) в пикселях.

height

[in] Высота (размер по оси Y) в пикселях.

clrfmt

[in] Способ обработки цвета. Более подробно о способах обработки цвета смотрите в описании функции ResourceCreate().

Возвращаемое значение

true в случае удачи, иначе — false.

© 2000-2025, MetaQuotes Ltd.

AllowedShowFlags

Устанавливает набор разрешенных флагов видимости для элементов графика.

```
void  AllowedShowFlags(
  const uint  flags,  // флаги
  )
```

Параметры

flags

[in] Разрешенные флаги.

© 2000-2025, MetaQuotes Ltd.

ShowLegend

Устанавливает значение флага видимости для легенды (FLAG_SHOW_LEGEND).

```
void  ShowLegend(
  const bool  flag,  // значение флага
  )
```

Параметры

flag

[in] Значение флага:

- true — легенда становится видимой.
- false — легенда становится невидимой.

© 2000-2025, MetaQuotes Ltd.

ShowScaleLeft

Устанавливает значение флага видимости для левой шкалы (FLAG_SHOW_SCALE_LEFT).

```
void  ShowScaleLeft(
  const bool  flag,  // значение флага
  )
```

Параметры

flag

[in] Значение флага:

- true — левая шкала становится видимой.
- false — левая шкала становится невидимой.

© 2000-2025, MetaQuotes Ltd.

ShowScaleRight

Устанавливает значение флага видимости для правой шкалы (FLAG_SHOW_SCALE_RIGHT).

```
void  ShowScaleRight(
  const bool  flag,  // значение флага
  )
```

Параметры

flag

[in] Значение флага:

- true — правая шкала становится видимой.
- false — правая шкала становится невидимой.

© 2000-2025, MetaQuotes Ltd.

ShowScaleTop

Устанавливает значение флага видимости для верхней шкалы (FLAG_SHOW_SCALE_TOP).

```
void  ShowScaleTop(
  const bool  flag,  // значение флага
  )
```

Параметры

flag

[in] Значение флага:

- true — верхняя шкала становится видимой.
- false — верхняя шкала становится невидимой.

© 2000-2025, MetaQuotes Ltd.

ShowScaleBottom

Устанавливает значение флага видимости для нижней шкалы (FLAG_SHOW_SCALE_BOTTOM).

```
void  ShowScaleBottom(
  const bool  flag,     // значение флага
  )
```

Параметры

flag

[in] Значение флага:

- true — нижняя шкала становится видимой.
- false — нижняя шкала становится невидимой.

© 2000-2025, MetaQuotes Ltd.

ShowGrid

Устанавливает значение флага видимости для сетки (FLAG_SHOW_GRID).

```
void  ShowGrid(
  const bool  flag,  // значение флага
  )
```

Параметры

flag

[in] Значение флага:

- true — сетка становится видимой.
- false — сетка становится невидимой.

© 2000-2025, MetaQuotes Ltd.

ShowDescriptors

Устанавливает значение флага видимости для дескрипторов (FLAG_SHOW_DESCRIPTORS).

```
void  ShowDescriptors(
  const bool  flag,    // значение флага
  )
```

Параметры

flag

[in] Значение флага:

- true — дескриптор становится видимым.
- false — дескриптор становится невидимым.

© 2000-2025, MetaQuotes Ltd.

ShowValue

Устанавливает флаг видимости для значений (FLAG_SHOW_VALUE).

```
void  ShowValue(
  const bool  flag,  // значение флага
  )
```

Параметры

flag

[in] Значение флага:

- true — значение становится видимым.
- false — значение становится невидимым.

© 2000-2025, MetaQuotes Ltd.

ShowPercent

Устанавливает значение флага видимости для процентов (FLAG_SHOW_PERCENT).

```
void  ShowPercent(
  const bool  flag,  // значение флага
  )
```

Параметры

flag

[in] Значение флага:

- true — процент становится видимым.
- false — процент становится невидимым.

© 2000-2025, MetaQuotes Ltd.

LegendAlignment

Устанавливает выравнивание текста для легенды.

```
void  LegendAlignment(
  const ENUM_ALIGNMENT  value,  // флаг
  )
```

Параметры

value

[in] Принимает одно из значений перечисления ENUM_ALIGNMENT:

- ALIGNMENT_LEFT — выравнивание по левому краю.
- ALIGNMENT_TOP — выравнивание по верхнему краю.
- ALIGNMENT_RIGHT — выравнивание по правому краю.
- ALIGNMENT_BOTTOM — выравнивание по нижнему краю.

© 2000-2025, MetaQuotes Ltd.

Accumulative

Устанавливает флаг накопления значений для серий.

```
void  Accumulative(
  const bool  flag=true,  // значение флага
  )
```

Параметры

flag=true

[in] Значение флага:

- true — текущее значение серии заменяется суммой всех предыдущих.
- false — стандартный режим отрисовки серий.

© 2000-2025, MetaQuotes Ltd.

VScaleParams

Устанавливает параметры для вертикальной шкалы значений.

```
void  VScaleParams(
  const double  max,   // максимум
  const double  min,   // минимум
  const uint    grid,  // количество делений
  )
```

Параметры

max

[in] Минимальное значение.

min

[in] Максимальное значение.

grid

[in] Количество делений шкалы.

© 2000-2025, MetaQuotes Ltd.

DescriptorUpdate

Обновляет значение дескриптора серии (по заданной позиции).

```
bool  DescriptorUpdate(
  const uint    pos,    // индекс
  const string  descr,  // значение
  )
```

Параметры

pos

[in] Индекс серии — порядковый номер её добавления, начниная с 0.

descr

[in] Значение дескриптора.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

ColorUpdate

Обновляет цвета серий (по заданной позиции).

```
bool  ColorUpdate(
  const uint  pos,  // индекс
  const uint  clr,  // цвет
  )
```

Параметры

pos

[in] Индекс серии — порядковый номер её добавления, начиная с 0.

clr

[in] Значение цвета.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

ValuesCheck

Вспомогательный виртуальный метод, производит внутренние вычисления для построения графика.

```
virtual void  ValuesCheck()
```

© 2000-2025, MetaQuotes Ltd.

Redraw

Виртуальный метод для перерисовывания графика.

```
virtual void  Redraw()
```

© 2000-2025, MetaQuotes Ltd.

DrawBackground

Виртуальный метод для перерисовки фона.

```
virtual void  DrawBackground()
```

© 2000-2025, MetaQuotes Ltd.

DrawLegend

Виртуальный метод для перерисовки легенды.

```
virtual void  DrawLegend()
```

© 2000-2025, MetaQuotes Ltd.

DrawLegendVertical

Рисует вертикальную легенду.

```
int  DrawLegendVertical(
  const int  w,  // ширина
  const int  h,  // высота
  )
```

Параметры

w

[in] Максимальная ширина текста в легенде.

h

[in] Максимальная высота текста в легенде.

Возвращаемое значение

Ширина легенды в пикселях.

© 2000-2025, MetaQuotes Ltd.

DrawLegendHorizontal

Рисует горизонтальную легенду.

```
int  DrawLegendHorizontal(
  const int  w,  //
  const int  h,  //
  )
```

Параметры

w

[in] Максимальная ширина текста в легенде.

h

[in] Максимальная высота текста в легенде.

Возвращаемое значение

Высота легенды в пикселях.

© 2000-2025, MetaQuotes Ltd.

CalcScales

Виртуальный метод для расчета координат надписей для шкалы значений.

```
virtual void  CalcScales()
```

© 2000-2025, MetaQuotes Ltd.

DrawScales

Виртуальный метод для перерисовки всех шкал значений.

```
virtual void  DrawScales()
```

© 2000-2025, MetaQuotes Ltd.

DrawScaleLeft

Виртуальный метод для перерисовки левой шкалы значений.

```
virtual int  DrawScaleLeft(
  const bool  draw,  // флаг
  )
```

Параметры

draw

[in] Флаг, указывающий, нужно ли выполнять перерисовку шкалы.

Возвращаемое значение

Ширина шкалы значений.

© 2000-2025, MetaQuotes Ltd.

DrawScaleRight

Виртуальный метод для перерисовки правой шкалы значений.

```
virtual int  DrawScaleRight(
  const bool  draw,  // флаг
  )
```

Параметры

draw

[in] Флаг, указывающий, нужно ли выполнять перерисовку шкалы.

Возвращаемое значение

Ширина шкалы значений.

© 2000-2025, MetaQuotes Ltd.

DrawScaleTop

Виртуальный метод для перерисовки верхней шкалы значений.

```
virtual int  DrawScaleTop(
  const bool  draw,  // флаг
  )
```

Параметры

draw

[in] Флаг, указывающий, нужно ли выполнять перерисовку шкалы.

Возвращаемое значение

Высота шкалы значений.

© 2000-2025, MetaQuotes Ltd.

DrawScaleBottom

Виртуальный метод для перерисовки нижней шкалы значений.

```
virtual int  DrawScaleBottom(
  const bool  draw,  // флаг
  )
```

Параметры

draw

[in] Флаг, указывающий, нужно ли выполнять перерисовку шкалы.

Возвращаемое значение

Высота шкалы значений.

© 2000-2025, MetaQuotes Ltd.

DrawGrid

Виртуальный метод для перерисовки сетки.

```
virtual void  DrawGrid()
```

© 2000-2025, MetaQuotes Ltd.

DrawChart

Виртуальный метод для перерисовки графика.

```
virtual void  DrawChart()
```

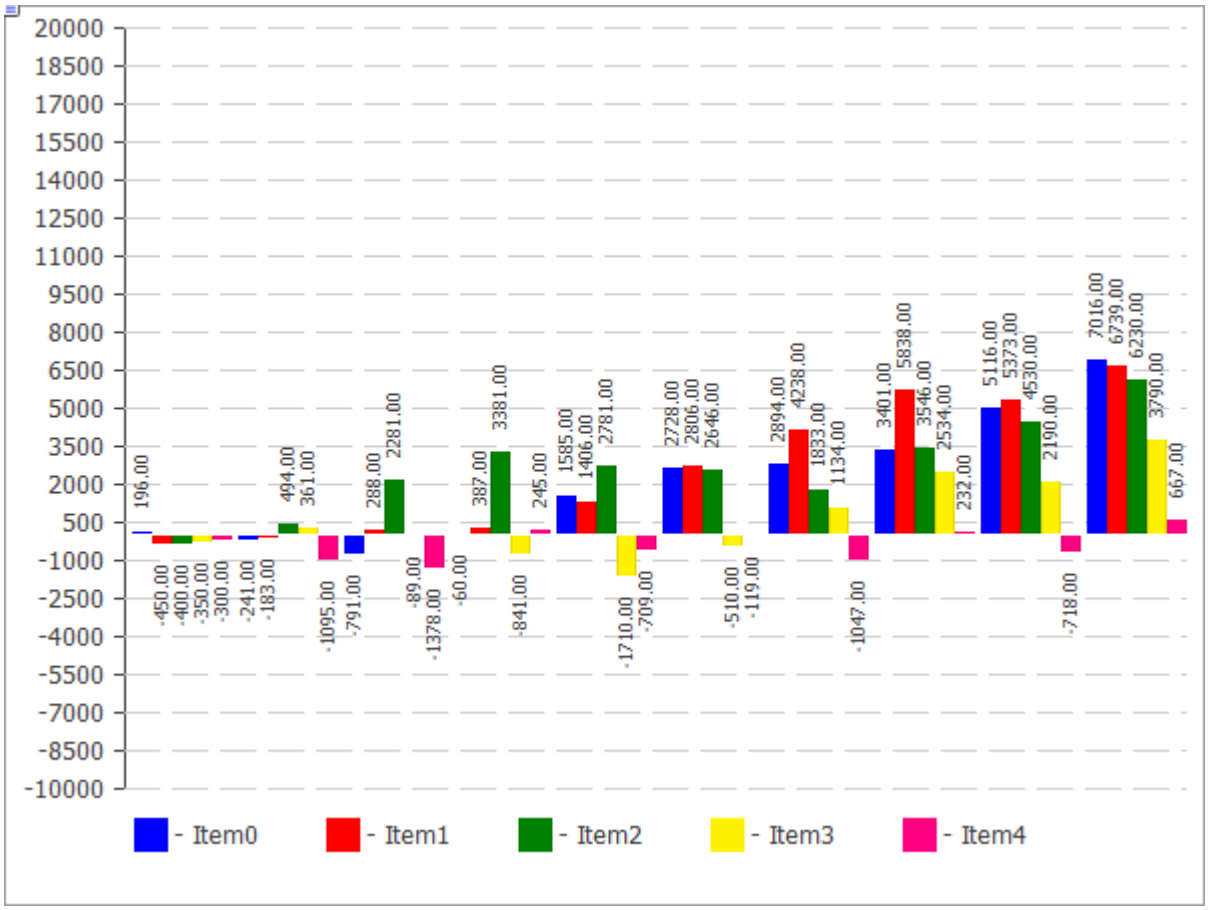
© 2000-2025, MetaQuotes Ltd.

CHistogramChart

Класс для построения гистограмм.

Описание

В этом классе реализованы все методы работы с построением столбчатых гистограмм. С их помощью задается ширина столбцов, настраивается работа с сериями данных. Включены методы для работы с градиентной заливкой столбцов гистограмм, что позволяет более наглядно отображать картину данных.



Код вышеприведенного рисунка представлен ниже.

Декларация

```
class CHistogramChart : public CChartCanvas
```

Заголовок

```
#include <Canvas\Charts\HistogramChart.mqh>
```

Иерархия наследования

CCanvas

CChartCanvas

CHistogramChart

© 2000-2025, MetaQuotes Ltd.

Методы класса

Метод	Действие
Gradient	Устанавливает флаг с указанием на то, будет ли применяться градиентная заливка столбцов гистограммы.
BarGap	Устанавливает значение отступа гистограммы от начала координат.
BarMinSize	Устанавливает минимальную ширину столбцов гистограммы.
BarBorder	Устаналивает флаг, указывающий на необходимость отрисовки границы для каждого столбца.
Create	Виртуальный метод, который создаёт графический ресурс.
SeriesAdd	Добавляет новую серию данных.
SeriesInsert	Вставляет серию данных на график.
SeriesUpdate	Обновляет серию данных на графике.
SeriesDelete	Удаляет серию данных на графике.
ValueUpdate	Обновляет значение элемента в указанной серии.
DrawData	Виртуальный метод, который рисует гистограмму для заданной серии.
DrawBar	Рисует столбец гистограммы в виде закрашенного прямоугольника.
GradientBrush	Создает кисть для градиентной заливки.

Методы унаследованные от 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

Методы унаследованные от CChartCanvas

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Пример

© 2000-2025, 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);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

Gradient

Устанавливает флаг с указанием на то, будет ли применяться градиентная заливка столбцов гистограммы.

```
void  Gradient(
  const bool  flag=true,  // значение флага
  )
```

Параметры

flag=true

Значение флага: true, если включена градиентная заливка, иначе — false.

© 2000-2025, MetaQuotes Ltd.

BarGap

Устанавливает значение отступа гистограммы от начала координат.

```
void  BarGap(
  const uint  value,  // отступ
  )
```

Параметры

value

[in] Значение отступа гистограммы.

© 2000-2025, MetaQuotes Ltd.

BarMinSize

Устанавливает минимальную ширину столбцов гистограммы.

```
void  BarMinSize(
  const uint  value,  // минимальная ширина
  )
```

Параметры

value

[in] Минимальная ширина.

© 2000-2025, MetaQuotes Ltd.

BarBorder

Устаналивает флаг, указывающий на необходимость отрисовки границы для каждого столбца.

```
void  BarBorder(
  const uint  value,  // флаг
  )
```

Параметры

value

[in] Значение флага:

- true — границы будут отрисовываться
- false — границы не будут отрисовываться

© 2000-2025, MetaQuotes Ltd.

Create

Виртуальный метод, который создаёт графический ресурс.

```
virtual bool  Create(
  const string       name,    // имя
  const int          width,   // ширина
  const int          height,  // высота
  ENUM_COLOR_FORMAT  clrfmt,  // формат
  )
```

Параметры

name

[in] Основание для имени графического ресурса. Имя ресурса формируется при создании путем добавления псевдослучайной строки.

width

[in] Ширина (размер по оси X) в пикселях.

height

[in] Высота (размер по оси Y) в пикселях.

clrfmt

[in] Способ обработки цвета. Более подробно о способах обработки цвета смотрите в описании функции ResourceCreate().

Возвращаемое значение

true в случае удачи, иначе — false.

© 2000-2025, MetaQuotes Ltd.

SeriesAdd

Добавляет новую серию данных.

```
bool  SeriesAdd(
  const double&  value[],  // значения
  const string   descr,    // подпись
  const uint     clr,      // цвет
  )
```

Параметры

value[]

[in] Серия данных.

descr

[in] Подпись серии.

clr

[in] Цвет отображения серии.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

SeriesInsert

Вставляет серию данных на график.

```
bool  SeriesInsert(
  const uint    pos,        // индекс
  const double&  value[],   // значения
  const string  descr,      // подпись
  const uint    clr,        // цвет
  )
```

Параметры

pos

[in] Индекс для вставки.

value[]

[in] Серия данных.

descr

[in] Подпись серии.

clr

[in] Цвет отображения серии.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

SeriesUpdate

Обновляет серию данных на графике.

```
bool  SeriesUpdate(
  const uint    pos,        // индекс
  const double  &value[],   // значения
  const string  descr,      // подпись
  const uint    clr,        // цвет
  )
```

Параметры

pos

[in] Индекс серии — порядковый номер её добавления, начиная с 0.

&value[]

[in] Новые значения для серии данных.

descr

[in] Подпись серии.

clr

[in] Цвет отображения серии.

Возвращаемое значение

true - в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

SeriesDelete

Удаляет серию данных на графике.

```
bool  SeriesDelete(
  const uint  pos,  // индекс
  )
```

Параметры

pos

[in] Индекс серии — порядковый номер её добавления, начниная с 0.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

ValueUpdate

Обновляет указанное значение в указанной серии.

```
bool  ValueUpdate(
  const uint   series,   // индекс серии
  const uint   pos,      // индекс элемента
  double       value,    // значение
  )
```

Параметры

series

[in] Индекс серии — порядковый номер её добавления, начниная с 0.

pos

[in] Индекс элемента в серии.

value

[in] Новое значение.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

DrawData

Виртуальный метод, который рисует гистограмму для заданной серии.

```
virtual void  DrawData(
  const uint  index,  // индекс
  )
```

Параметры

index

[in] Индекс серии — порядковый номер её добавления, начиная с 0.

© 2000-2025, MetaQuotes Ltd.

DrawBar

Рисует столбец гистограммы в виде закрашенного прямоугольника.

```
void  DrawBar(
  const int   x,    // координата x
  const int   y,    // координата y
  const int   w,    // ширина
  const int   h,    // высота
  const uint  clr,  // цвет
  )
```

Параметры

x

[in] Координата X левой верхней точки прямоугольника.

y

[in] Координата Y левой верхней точки прямоугольника.

w

[in] Ширина прямоугольника.

h

[in] Высота прямоугольника.

clr

[in] Цвет прямоугольника.

© 2000-2025, MetaQuotes Ltd.

GradientBrush

Создает кисть для градиентной заливки.

```
void  GradientBrush(
  const int   size,      // размер
  const uint  fill_clr,  // цвет заливки
  )
```

Параметры

size

[in] Толщина кисти

fill_clr

[in] Цвет заливки

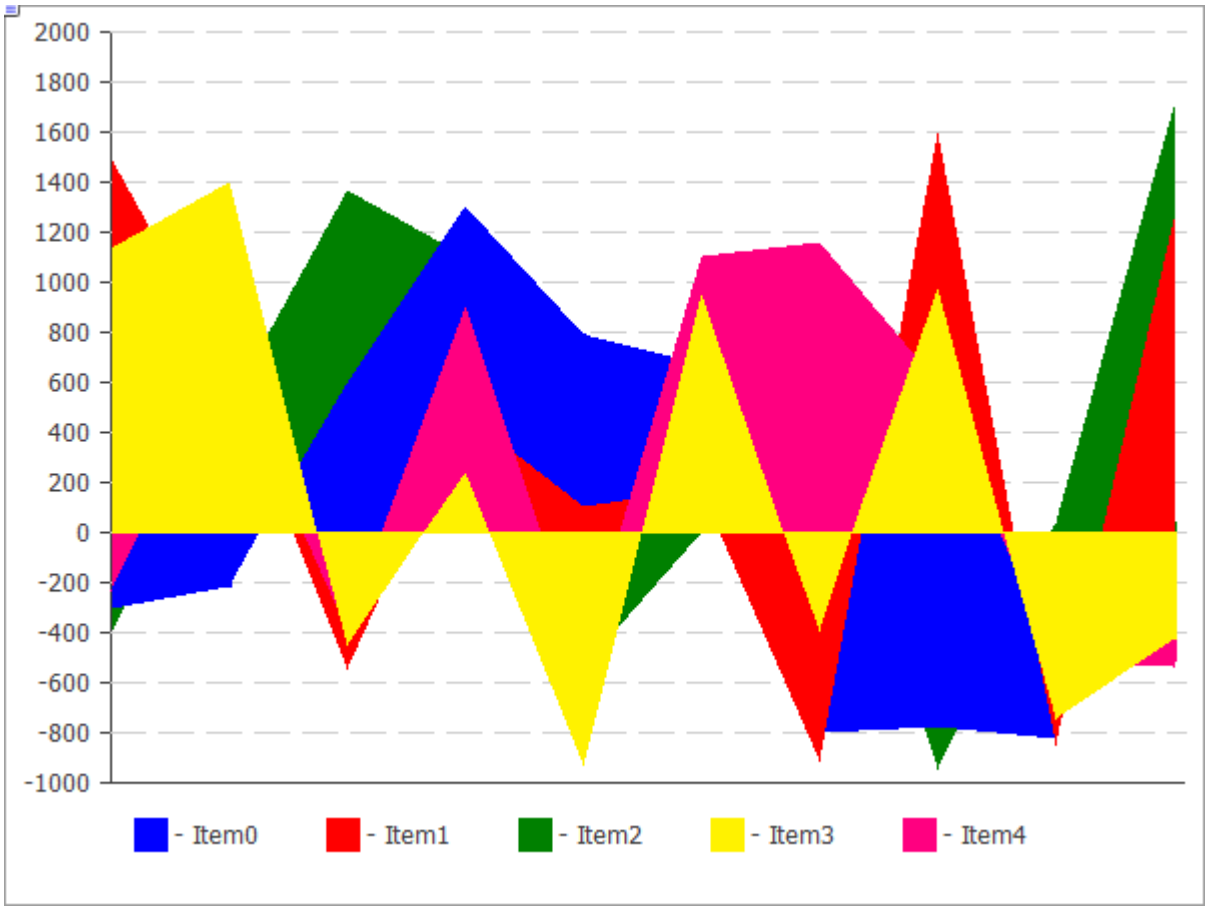
© 2000-2025, MetaQuotes Ltd.

CLineChart

Класс для построения кривых.

Описание

Методы, входящие в класс, предназначены для работы с кривыми на графике. Включена возможность заливки области, ограниченной построенной кривой.



Код вышеприведенного рисунка представлен ниже.

Декларация

```
class CLineChart : public CChartCanvas
```

Заголовок

```
#include <Canvas\Charts\LineChart.mqh>
```

Иерархия наследования

CCanvas
  CChartCanvas
    CLineChart

Методы класса

© 2000-2025, MetaQuotes Ltd.

Method	Action
Filled	Устанавливает флаг заливки области под кривой, заданной серией данных.
Create	Создает графический ресурс.
SeriesAdd	Добавляет новую серию данных.
SeriesInsert	Вставляет серию данных на график.
SeriesUpdate	Обновляет серию данных на графике.
SeriesDelete	Удаляет серию данных с графика.
ValueUpdate	Обновляет указанное значение в указанной серии.
DrawChart	Виртуальный метод, который выполняет отрисовку кривой и всех её элементов.
DrawData	Виртуальный метод, который рисует кривую для заданной серии.
CalcArea	Рассчитывает площадь под кривой, заданной серией данных.

Методы унаследованные от 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

Методы унаследованные от 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

Пример

© 2000-2025, 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);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

Filled

Устанавливает флаг с указанием, нужно ли закрашивать облать под кривой, заданной серией данных.

```
void  Filled(
  const bool  flag=true,  // флаг
  )
```

Параметры

flag=true

[in] Значение флага:

- true — закрашивать область под кривой
- false — не закрашивать область под кривой

© 2000-2025, MetaQuotes Ltd.

Create

Виртуальный метод, который создаёт графический ресурс.

```
virtual bool  Create(
  const string       name,    // имя
  const int          width,   // ширина
  const int          height,  // высота
  ENUM_COLOR_FORMAT  clrfmt,  // формат
  )
```

Параметры

name

[in] Основание для имени графического ресурса. Имя ресурса формируется при создании путем добавления псевдослучайной строки.

width

[in] Ширина (размер по оси X) в пикселях.

height

[in] Высота (размер по оси Y) в пикселях.

clrfmt

[in] Способ обработки цвета. Более подробно о способах обработки цвета смотрите в описании функции ResourceCreate().

Возвращаемое значение

true в случае удачи, иначе — false.

© 2000-2025, MetaQuotes Ltd.

SeriesAdd

Добавляет новую серию данных.

```
bool  SeriesAdd(
  const double&  value[],  // значения
  const string   descr,    // подпись
  const uint     clr,      // цвет
  )
```

Параметры

value[]

[in] Серия данных.

descr

[in] Подпись серии.

clr

[in] Цвет отображения серии.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

SeriesInsert

Вставляет серию данных на график.

```
bool  SeriesInsert(
  const uint     pos,        // индекс
  const double&  value[],    // значения
  const string   descr,      // подпись
  const uint     clr,        // цвет
  )
```

Параметры

pos

[in] Индекс для вставки.

value[]

[in] Серия данных.

descr

[in] Подпись серии.

clr

[in] Цвет отображения серии.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

SeriesUpdate

Обновляет серию данных на графике.

```
bool  SeriesUpdate(
  const uint     pos,        // индекс
  const double&  value[],    // значения
  const string   descr,      // подпись
  const uint     clr,        // цвет
  )
```

Параметры

pos

[in] Индекс серии — порядковый номер её добавления, начиная с 0.

value[]

[in] Новые значения для серии данных.

descr

[in] Подпись серии.

clr

[in] Цвет отображения серии.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

SeriesDelete

Удаляет серию данных с графика.

```
bool  SeriesDelete(
  const uint  pos,  // индекс
  )
```

Параметры

pos

[in] Индекс серии — порядковый номер её добавления, начиная с 0.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

ValueUpdate

Обновляет указанное значение в указанной серии.

```
bool  ValueUpdate(
  const uint   series,   // индекс серии
  const uint   pos,      // индекс элемента
  double       value,    // значение
  )
```

Параметры

series

[in] Индекс серии — порядковый номер её добавления, начниная с 0.

pos

[in] Индекс элемента в серии.

value

[in] Новое значение.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

DrawChart

Виртуальный метод, который выполняет отрисовку кривой и всех её элементов.

```
virtual void  DrawChart()
```

© 2000-2025, MetaQuotes Ltd.

DrawData

Виртуальный метод, который рисует кривую для заданной серии.

```
virtual void  DrawData(
  const uint  index,  // индекс
  )
```

Parameters

index

[in] Индекс серии — порядковый номер её добавления, начиная с 0.

© 2000-2025, MetaQuotes Ltd.

CalcArea

Рассчитывает площадь под кривой, заданной серией данных.

```
double  CalcArea(
  const uint  index,  // индекс
  )
```

Параметры

index

[in] Индекс серии — порядковый номер её добавления, начиная с 0.

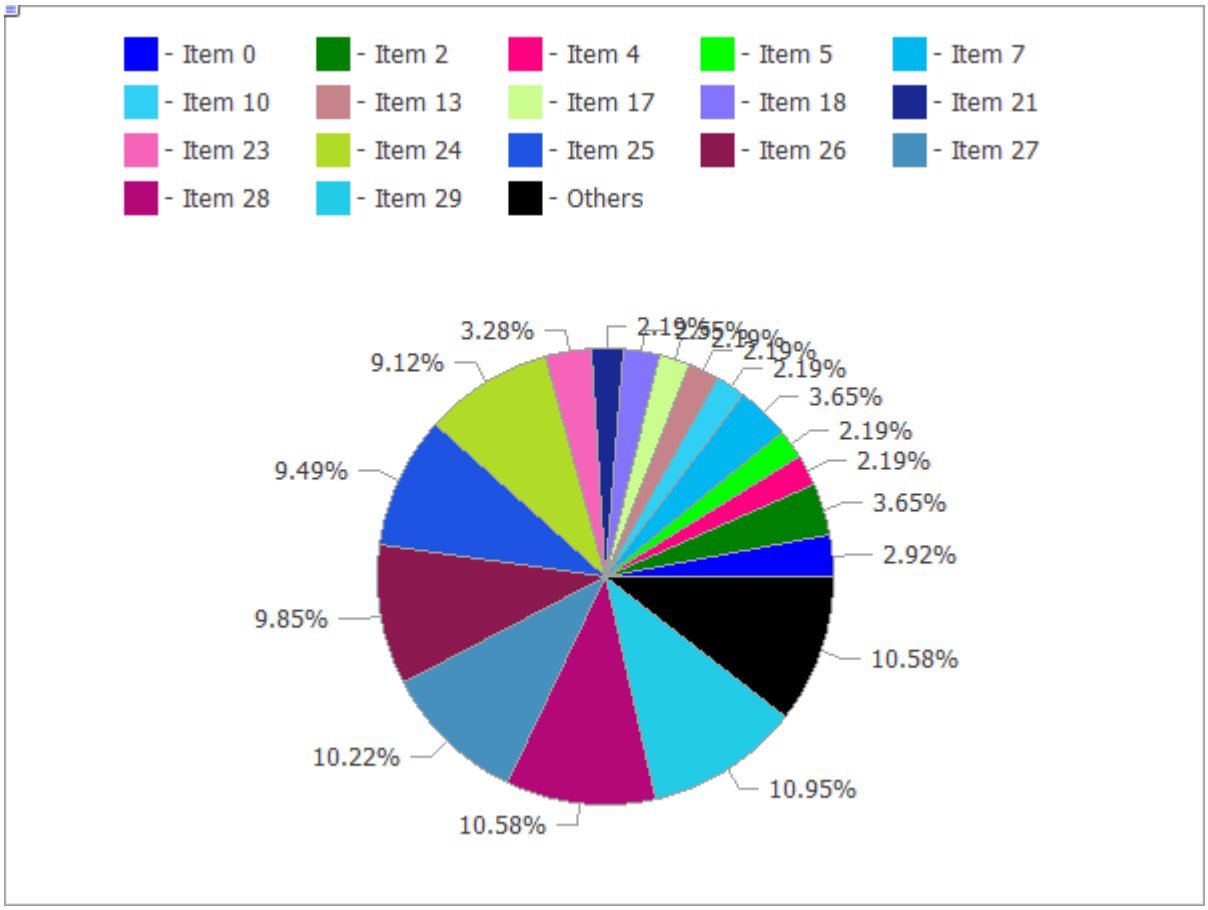
Возвращаемое значение

Площадь фигуры, которую ограничивает кривая, заданная серией данных.

© 2000-2025, MetaQuotes Ltd.

CPieChart

Класс для построения круговых диаграмм.



Код вышеприведенного рисунка представлен ниже.

Описание

Методы, входящие в этот класс, предназначены для полноценной работы с круговыми диаграммами, начиная с создания графического ресурса и заканчивая оформлением подписей к сегментам.

Декларация

```
class CPieChart : public CChartCanvas
```

Заголовок

```
#include <Canvas\Charts\PieChart.mqh>
```

Иерархия наследования

CCanvas

CChartCanvas

CPieChart

© 2000-2025, MetaQuotes Ltd.

Методы класса

Метод	Действие
Create	Виртуальный метод, который создаёт графический ресурс.
SeriesSet	Задает серию значений, которые будут показаны на диаграмме.
ValueAdd	Добавляет новое значение на диаграмму (в конец).
ValueInsert	Вставляет новое значение на диаграмму (по указанной позиции).
ValueUpdate	Обновляет значение на диаграмме (по указанной позиции).
ValueDelete	Удаляет значение с диаграммы (по заданной позиции).
DrawChart	Виртуальный метод, который выполняет отрисовку диаграммы и всех её элементов.
DrawPie	Рисует сегмент диаграммы, соответствующий определенному значению.
LabelMake	Генерирует подпись сегмента на основе его значения и изначальной подписи.

Методы унаследованные от 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

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Пример

```
//+------------------------------------------------------------------+
//|                                                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);
```

© 2000-2025, MetaQuotes Ltd.

```
   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));
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Create

Виртуальный метод, который создаёт графический ресурс.

```
virtual bool  Create(
  const string       name,    // имя
  const int          width,   // ширина
  const int          height,  // высота
  ENUM_COLOR_FORMAT  clrfmt,  // формат
  )
```

Параметры

name

[in] Основание для имени графического ресурса. Имя ресурса формируется при создании путем добавления псевдослучайной строки.

width

[in] Ширина (размер по оси X) в пикселях.

height

[in] Высота (размер по оси Y) в пикселях.

clrfmt

[in] Способ обработки цвета. Более подробно о способах обработки цвета смотрите в описании функции ResourceCreate().

Возвращаемое значение

true в случае удачи, иначе — false.

© 2000-2025, MetaQuotes Ltd.

SeriesSet

Задает серию значений, которые будут показаны на диаграмме.

```
bool  SeriesSet(
  const double&  value[],  // значения
  const string&  text[],   // надписи
  const uint&    clr[],    // цвета
  )
```

Параметры

value[]

[in] Массив значений.

text[]

[in] Массив подписей значений.

clr[]

[in] Массив цветов значений.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

ValueAdd

Добавляет новое значение на диаграмму (в конец).

```
bool  ValueAdd(
  const double  value,  // значение
  const string  descr,  // подпись
  const uint    clr,    // цвет
  )
```

Параметры

value

[in] Значение.

descr

[in] Подпись значения.

clr

[in] Цвет значения.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

ValueInsert

Вставляет новое значение на диаграмму (по указанной позиции).

```
bool  ValueInsert(
  const uint    pos,    // индекс
  const double  value,  // значение
  const string  descr,  // подпись
  const uint    clr,    // цвет
  )
```

Параметры

pos

[in] Индекс для вставки.

value

[in] Значение.

descr

[in] Подпись значения.

clr

[in] Цвет значения.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

ValueUpdate

Обновляет значение на диаграмме (по указанной позиции).

```
bool  ValueUpdate(
  const uint    pos,    // индекс
  const double  value,  // значение
  const string  descr,  // подпись
  const uint    clr,    // цвет
  )
```

Параметры

pos

[in] Индекс значения — порядковый номер его добавления, начниная с 0.

value

[in] Значение.

descr

[in] Подпись значения.

clr

[in] Цвет значения.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

ValueDelete

Удаляет значение с диаграммы (по заданной позиции).

```
bool  ValueDelete(
  const uint  pos,  // индекс
  )
```

Параметры

pos

[in] Индекс значения — порядковый номер его добавления, начниная с 0.

Возвращаемое значение

true в случае успеха, иначе — false.

© 2000-2025, MetaQuotes Ltd.

DrawChart

Виртуальный метод, который выполняет отрисовку диаграммы и всех её элементов.

```
virtual void  DrawChart()
```

© 2000-2025, MetaQuotes Ltd.

DrawPie

Рисует сегмент диаграммы, соответствующий определенному значению.

```
void  DrawPie(
  double      fi3,   // угол луча из центра диаграммы, задающий первую границу дуги
  double      fi4,   // угол луча из центра диаграммы, задающий вторую границу дуги
  int         idx,   // индекс
  CPoint&     p[],  //
  const uint  clr,   //
  )
```

Параметры

fi3

[in] Угол в радианах, задающий первую границу дуги.

fi4

[in] Угол в радианах, задающий вторую границу дуги.

idx

[in] Индекс значения, которому соответствует сегмент.

p[]

[in] Массив опорных точек (x, y) для построения сегментов.

clr

[in] Цвет сегмента.

© 2000-2025, MetaQuotes Ltd.

LabelMake

Генерирует подпись сегмента на основе его значения и изначальной подписи.

```
string  LabelMake(
  const string  text,      // подпись
  const double  value,     // значение
  const bool    to_left,   // флаг
  )
```

Параметры

text

[in] Подпись.

value

[in] Значение.

to_left

[in] Определяет порядок компоновки подписи:

- true — подпись, затем значение.
- false — значение, затем подпись.

Возвращаемое значение

Подпись сегмента.

© 2000-2025, MetaQuotes Ltd.

3D графика

В этом разделе представлены классы создания трехмерной графики, которые построены на основе функций для работы с DirectX. Базовым классом является CCnavas3D, который содержит методы управления камерой и освещением, а также предоставляет менеджер графических ресурсов – текстур, шейдеров, буферов вершин, индексов и параметров шейдеров.

Для того чтобы начать работать с библиотекой, достаточно прочитать статью Как создать 3D-графику на DirectX в MetaTrader 5.

Отдельно представлены классы базовых объектов сцены, такие как параллелепипед, трехмерная поверхность на пользовательских данных и произвольная сетка.

© 2000-2025, MetaQuotes Ltd.

CCanvas3D

Класс CCanvas3D является классом для упрощенного создания и визуализации трехмерных объектов на графике.

Описание

CCanvas3D существенно упрощает создание и визуализацию больших объемов данных в виде анимированной 3D графики. Класс содержит методы управления камерой и освещением, а также предоставляет менеджер ресурсов для создания графических ресурсов: текстур, шейдеров, буферов вершин, индексов и параметров шейдеров.

Кроме того, библиотека содержит классы базовых объектов сцены, такие как параллелепипед, трехмерная поверхность на пользовательских данных либо произвольная сетка.

Для работы функций видеокарта должна поддерживать DX 11 и шейдеры версии 5.0.

Декларация

```
class CCanvas
```

Заголовок

```
#include <Canvas\Canvas.mqh>
```

Иерархия наследования

CCanvas

CCanvas3D

Методы класса по группам

Создание/удаление	Описание
Create	Создает графический ресурс для отрисовки 3D-сцены без привязки к объекту чарта.
Attach	Получает из объекта OBJ_BITMAP_LABEL графический ресурс и привязывает его к экземпляру класса CCanvas.
ObjectAdd	Добавляет на 3D-сцену объект для последующей отрисовки.
Destroy	Уничтожает графический ресурс и освобождает графический 3D-контекст.
Работа со светом	
AmbientColorSet	Устанавливает цвет и интенсивность окружающего ненаправленного освещения.
AmbientColorGet	Получает цвет и интенсивность окружающего ненаправленного освещения.
LightDirectionSet	Задает направление направленного источника света.

© 2000-2025, MetaQuotes Ltd.

Создание/удаление	Описание
LightDirectionGet	Получает направление направленного источника света.
LightColorSet	Устанавливает цвет и интенсивность направленного источника света.
LightColorGet	Получает цвет и интенсивность направленного источника света.
Работа с камерой	
ProjectionMatrixSet	Вычисляет и устанавливает матрицу проекции 3D-координаты на 2D кадр.
ProjectionMatrixGet	Получает матрицу проекции 3D-сцены на 2D кадр.
ViewMatrixSet	Устанавливает матрицу вида 3D-сцены.
ViewMatrixGet	Возвращает матрицу вида 3D-сцены.
ViewPositionSet	Устанавливает положение точки зрения на 3D-сцену.
ViewRotationSet	Задает направление взгляда на 3D-сцену.
ViewTargetSet	Устанавливает координаты точки, на которую направлен взгляд.
ViewUpDirectionSet	Устанавливает направление верхней границы кадра в 3D-пространстве.
Отрисовка	
Render	Выполняет полный цикл отрисовки всех объектов сцены во внутреннем буфере кадра для последующего отображения.
RenderBegin	Подготавливает графический контекст для отрисовки нового кадра.
RenderEnd	Копирует отрисованный кадр во внутренний буфер и обновляет при необходимости изображение на чарте.
Получение ресурсов	
DXContext	Возвращает хендл графического контекста.
DXDispatcher	Возвращает хендл диспетчера ресурсов.
InputScene	Возвращает указатель на буфер параметров сцены.

© 2000-2025, MetaQuotes Ltd.

AmbientColorGet

Получает цвет и интенсивность окружающего ненаправленного освещения.

```
void  AmbientColorGet(
   DXColor  &ambient_color      // цвет и интенсивность ненаправленного освещения
   );
```

Параметры

&ambient_color

[out] Цвет ненаправленного освещения.

Возвращаемое значение

Нет

Примечание

Интенсивность хранится в альфа-канале структуры DXColor.

© 2000-2025, MetaQuotes Ltd.

AmbientColorSet

Устанавливает цвет и интенсивность окружающего ненаправленного освещения.

```
void  AmbientColorSet(
   const DXColor  &ambient_color      // цвет и интенсивность ненаправленного освещени
   );
```

Параметры

&ambient_color

[in] Цвет ненаправленного освещения.

Возвращаемое значение

Нет

Примечание

Интенсивность задается в альфа-канале структуры DXColor.

© 2000-2025, MetaQuotes Ltd.

Attach

Получает из объекта OBJ_BITMAP_LABEL графический ресурс и привязывает его к экземпляру класса CCanvas.

```
bool  Attach(
   const long          chart_id,                          // идентификатор чарта
   const string        objname,                           // имя объекта
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // способ обработки цвета
   )
```

Создает графический ресурс для объекта OBJ_BITMAP_LABEL и привязывает его к экземпляру класса CCanvas.

```
bool  Attach(
   const long          chart_id,                          // идентификатор чарта
   const string        objname,                           // имя объекта
   const int           width,                             // ширина картинки в пик
   const int           height,                            // высота картинки в пик
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // способ обработки цвета
   )
```

Параметры

chart_id

[in] Идентификатор графика.

objname

[in] Наименование (имя) графического объекта.

width

[in] Ширина кадра в ресурсе.

height

[in] Высота кадра.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Способ обработки цвета. Более подробно о способах обработки цвета смотрите в описании функции ResourceCreate().

Примечание

true - в случае удачи, false - если не удалось добавить графический объект.

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический ресурс для отрисовки 3D-сцены без привязки к объекту чарта.

```
virtual bool  Create(
   const string       name,                                  // имя графического объект
   const int          width,                                 // ширина
   const int          height,                                // высота
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // формат цвета
   );
```

Параметры

name

[in] Имя графического объекта.

width

[in] Ширина кадра.

height

[in] Высота кадра.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Способ обработки цвета. Более подробно о способах обработки цвета смотрите в описании функции ResourceCreate().

Примечание

true при успешном создании ресурса, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Destroy

Уничтожает графический ресурс и освобождает графический 3D-контекст.

```
virtual void  Destroy()
```

Возвращаемое значение

Нет

Примечание

Если графический ресурс создавался с привязкой к объекту чарта, то объект чарта удаляется.

© 2000-2025, MetaQuotes Ltd.

DXContext

Возвращает хендл графического контекста.

```
int  DXContext()
```

Возвращаемое значение

Хендл графического контекста.

© 2000-2025, MetaQuotes Ltd.

DXDispatcher

Возвращает хендл диспетчера ресурсов.

```
CDXDispatcher*  DXDispatcher()
```

Возвращаемое значение

Хендл диспетчера ресурсов.

© 2000-2025, MetaQuotes Ltd.

InputScene

Возвращает указатель на буфер параметров сцены.

```
CDXInput*  InputScene()
```

Возвращаемое значение

Указатель на буфер параметров сцены.

© 2000-2025, MetaQuotes Ltd.

LightColorGet

Получает цвет и интенсивность направленного источника света.

```
void  LightColorGet(
   DXColor  &light_color      // цвет и интенсивность направленного освещения
   );
```

Параметры

&light_color

[out] Цвет и интенсивность направленного освещения.

Возвращаемое значение

Нет.

Примечание

Интенсивность хранится в альфа-канале структуры DXColor.

© 2000-2025, MetaQuotes Ltd.

LightColorSet

Устанавливает цвет и интенсивность направленного источника света.

```
void  LightColorSet(
   const DXColor  &light_color      // цвет и интенсивность направленного освещения
   );
```

Параметры

&light_color

[in] Цвет и интенсивность направленного освещения.

Возвращаемое значение

Нет.

Примечание

Интенсивность задается в альфа-канале структуры DXColor.

© 2000-2025, MetaQuotes Ltd.

LightDirectionGet

Получает направление направленного источника света.

```
void  LightDirectionGet(
   DXVector3  &light_direction      // вектор направления
   );
```

Параметры

&light_direction

[out] Вектор направления.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

LightDirectionSet

Задает направление направленного источника света.

```
void  LightDirectionSet(
   const DXVector3  &light_direction      // вектор направления
   );
```

Параметры

&light_direction

[in] Вектор направления.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

ObjectAdd

Добавляет на 3D-сцену объект для последующей отрисовки.

```
bool  ObjectAdd(
   CDXObject  *object      // указатель на объект
   );
```

Параметры

object

[in] Указатель на экземпляр класса, являющегося наследником абстрактного класса CDXObject.

Возвращаемое значение

true - в случае удачи, false - если не удалось добавить графический 3D-объект.

© 2000-2025, MetaQuotes Ltd.

ProjectionMatrixGet

Получает матрицу проекции 3D-сцены на 2D кадр.

```
void  ProjectionMatrixGet(
   DXMatrix  &projection_matrix      // матрица проекции
   );
```

Параметры

&projection_matrix

[out] Матрица проекции.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

ProjectionMatrixSet

Вычисляет и устанавливает матрицу проекции 3D-координаты на 2D кадр.

```
void  ProjectionMatrixSet(
   float  fov,              // угол зрения (field of view)
   float  aspect_ratio,     // отношение сторон кадра
   float  z_near,           //
   float  z_far             //
   );
```

Параметры

fov

[in] Ширина поля зрения в радианах для создания проекции сцены.

aspect_ratio

[in] Соотношение сторон 2D-кадра.

z_near

[in] Расстояние до ближней плоскости отсечения.

z_far

[in] Расстояние до дальней плоскости отсечения.

Возвращаемое значение

Нет.

Примечание

На 2D-кадре будут показаны проекции только тех 3D-объектов, которые попадают в указанное поле зрения и расположены между ближней и дальней плоскостями отсечения.

© 2000-2025, MetaQuotes Ltd.

Render

Выполняет полный цикл отрисовки всех объектов сцены во внутреннем буфере кадра для последующего отображения.

```
bool  Render(
   uint  flags,                  // комбинация флагов
   uint  background_color=0      // цвет фона
   );
```

Параметры

flags

[in] Комбинация флагов, задающая режим отрисовки. Возможные значения:
DX_CLEAR_COLOR - очистить буфер изображения цветом *background_color*.
DX_CLEAR_DEPTH - очистить буфер глубины.

background_color=0

[in] Цвет фона 3D-сцены.

Возвращаемое значение

true - в случае удачи, false - если не удалось отрисовать.

Примечание

Вызов Render() не приводит к обновлению сцены на графике, а только обновляет внутренний буфер изображения. Для отрисовки обновленного кадра нужно явно вызвать метод Update().

Render() содержит в себе вызовы RenderBegin и RenderEnd().

© 2000-2025, MetaQuotes Ltd.

RenderBegin

Подготавливает графический контекст для отрисовки нового кадра.

```
virtual bool  RenderBegin(
   uint  flags,                  // комбинация флагов
   uint  background_color=0      // цвет фона
   );
```

Параметры

flags

[in] Комбинация флагов, задающая режим отрисовки. Возможные значения:
DX_CLEAR_COLOR - очистить буфер изображения цветом *background_color*.
DX_CLEAR_DEPTH - очистить буфер глубины.

background_color=0

[in] Цвет фона 3D-сцены.

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить входные параметры шейдера.

© 2000-2025, MetaQuotes Ltd.

RenderEnd

Копирует отрисованный кадр во внутренний буфер и обновляет при необходимости изображение на чарте.

```
virtual bool  RenderEnd(
   bool  redraw=false      // флаг обновления
   );
```

Параметры

redraw=false

[in] Флаг необходимости перерисовки чарта.

Возвращаемое значение

true - в случае удачи, иначе false.

© 2000-2025, MetaQuotes Ltd.

ViewMatrixGet

Возвращает матрицу вида 3D-сцены.

```
void  ViewMatrixGet(
   DXMatrix  &view_matrix      // матрица вида
   );
```

Параметры

&view_matrix

[out] Матрица вида, задающая положение и направление камеры в 3D-пространстве.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

ViewMatrixSet

Устанавливает матрицу вида 3D-сцены.

```
void  ViewMatrixSet(
   const DXMatrix  &view_matrix      // матрица вида
   );
```

Параметры

&view_matrix

[in] Матрица вида, задающая положение и направление камеры в 3D-пространстве.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

ViewPositionSet

Устанавливает положение точки зрения на 3D-сцену.

```
void  ViewPositionSet(
   const DXVector3  &position      // положение точки зрения
   );
```

Параметры

&posiiton

[in] Положение точки зрения на 3D-сцену.

Возвращаемое значение

Нет.

Примечание

Установка положения точки зрения с помощью ViewPositionSet() изменяет матрицу вида, получаемую в ViewMatrixGet().

© 2000-2025, MetaQuotes Ltd.

ViewRotationSet

Задает направление взгляда на 3D-сцену.

```
void  ViewRotationSet(
   const DXVector3  &rotation      // вектор углов поворота
   );
```

Параметры

&rotation

[in] Вектор, задающий углы Эйлера для вычисления направления взгляда на 3D-сцену.

Возвращаемое значение

Нет.

Примечание

Установка направления взгляда с помощью ViewRotationSet() изменяет матрицу вида, получаемую в ViewMatrixGet().

© 2000-2025, MetaQuotes Ltd.

ViewTargetSet

Устанавливает координаты точки, на которую направлен взгляд.

```
void  ViewTargetSet(
   const DXVector3  &target      // координаты цели
   );
```

Параметры

&target

[in] Координаты точки, на которую направлен взгляд.

Возвращаемое значение

Нет.

Примечание

Используется для фиксации взгляда на одной точке сцены при перемещении точки зрения.

Установка новой координаты цели с помощью ViewRotationSet() изменяет матрицу вида, получаемую в ViewMatrixGet().

ViewTargetSet() используется совместно с ViewUpDirectionSet() для определения направления взгляда.

© 2000-2025, MetaQuotes Ltd.

ViewUpDirectionSet

Устанавливает направление верхней границы кадра в 3D-пространстве.

```
void  ViewUpDirectionSet(
   const DXVector3  &up_direction      // направление верха
   );
```

Параметры

&up_direction

[in] Направление верха кадра в 3D-пространстве.

Возвращаемое значение

Нет.

Примечание

Установка нового направления с помощью ViewUpDirectionSet() изменяет матрицу вида, получаемую в ViewMatrixGet().

ViewUpDirectionSet() используется совместно с ViewTargetSet() для определения направления взгляда.

© 2000-2025, MetaQuotes Ltd.

CChart

Класс CChart является классом для упрощенного доступа к свойствам графика.

Описание

Класс CChart обеспечивает доступ к свойствам графика.

Декларация

```
class CChart : public CObject
```

Заголовок

```
#include <Charts\Chart.mqh>
```

Иерархия наследования

CObject

CChart

Методы класса по группам

Доступ к защищенным данным	
ChartID	Получает идентификатор графика
Общие свойства	
Mode	Получить/установить тип графика (свечи, бары или линия)
Foreground	Получить/установить флаг "Ценовой график на заднем фоне"
Shift	Получить/установить флаг "Режим отступа ценового графика от правого края"
ShiftSize	Получить/установить размер отступа нулевого бара от правого края в процентах
AutoScroll	Получить/установить флаг "Режим автоматического перехода к правому краю графика"
Scale	Получить/установить масштаб
ScaleFix	Получить/установить флаг "Режим фиксированного масштаба"
ScaleFix_11	Получить/установить флаг "Режим масштаба 1:1"
FixedMax	Получить/установить фиксированный максимум графика
FixedMin	Получить/установить фиксированный минимум графика
ScalePPB	Получить/установить флаг "Режим указания масштаба в пунктах на бар"

© 2000-2025, MetaQuotes Ltd.

Доступ к защищенным данным	
PointsPerBar	Получить/установить масштаб в пунктах на бар
Свойства "Show"	
ShowOHLC	Получить/установить флаг "Отображение значений OHLC"
ShowLineBid	Получить/установить флаг "Отображение значения Bid горизонтальной линией"
ShowLineAsk	Получить/установить флаг "Отображение значения Ask горизонтальной линией"
ShowLastLine	Получить/установить флаг "Отображение значения Last горизонтальной линией"
ShowPeriodSep	Получить/установить флаг "Отображение вертикальных разделителей между соседними периодами"
ShowGrid	Получить/установить флаг "Отображение сетки"
ShowVolumes	Получить/установить флаг "Отображение объемов"
ShowObjectDescr	Получить/установить флаг "Отображение всплывающих описаний графических объектов"
ShowDateScale	Установить флаг "Отображение шкалы времени"
ShowPriceScale	Установить флаг "Отображение ценовой шкалы"
Свойства "Color"	
ColorBackground	Получить/установить цвет фона
ColorForeground	Получить/установить цвет осей, шкалы и строки OHLC
ColorGrid	Получить/установить цвет сетки
ColorBarUp	Получить/установить цвет бара вверх, тени и окантовки тела бычьей свечи
ColorBarDown	Получить/установить цвет бара вниз, тени и окантовки тела медвежьей свечи
ColorCandleBull	Получить/установить цвет тела бычьей свечи
ColorCandleBear	Получить/установить цвет тела медвежьей свечи
ColorChartLine	Получить/установить цвет линии графика и японских свечей "Доджи"
ColorVolumes	Получить/установить цвет объемов и уровней открытия позиций
ColorLineBid	Получить/установить цвет линии Bid-цены
ColorLineAsk	Получить/установить цвет линии Ask-цены

© 2000-2025, MetaQuotes Ltd.

Доступ к защищенным данным	
ColorLineLast	Получить/установить цвет линии цены последней совершенной сделки
ColorStopLevels	Получить/установить цвет уровней стоп-ордеров (Stop Loss и Take Profit)
Свойства "Read only"	
VisibleBars	Получить количество баров на графике, доступных для отображения
WindowsTotal	Получить общее количество окон графика, включая подокна индикаторов
WindowIsVisible	Получить видимость подокон
WindowHandle	Получить хэндл графика (HWND)
FirstVisibleBar	Получить номер первого видимого бара на графике
WidthInBars	Получить ширину графика в барах
WidthInPixels	Получить ширину графика в пикселях
HeightInPixels	Получить высоту окна в пикселях
PriceMin	Получить минимум окна
PriceMax	Получить максимум окна
Свойства	
Attach	Привязывает график к экземпляру класса
FirstChart	Получить идентификатор первого графика
NextChart	Получить идентификатор графика, следующего за указанным
Open	Открыть новый график
Detach	Отвязывает график от экземпляра класса
Close	Закрыть указанный график
BringToTop	Показывает график поверх всех других
EventObjectCreate	Устанавливает флаг отправки сообщений о событиях создания графического объекта
EventObjectDelete	Устанавливает флаг отправки сообщений о событиях удаления графического объекта
Индикаторы	
IndicatorAdd	Добавляет на указанное окно графика индикатор с указанным хэндлом

© 2000-2025, MetaQuotes Ltd.

Доступ к защищенным данным	
IndicatorDelete	Удаляет с указанного окна графика индикатор с указанным именем
IndicatorsTotal	Возвращает количество всех индикаторов, присоединенных к указанному окну графика
IndicatorName	Возвращает короткое имя индикатора по номеру в списке индикаторов на указанном окна графика
Навигация	
Navigate	Сдвиг графика
Доступ к API MQL5	
Symbol	Получить имя символа графика
Period	Получить значение периода графика
Redraw	Принудительная перерисовка графика
GetInteger	Получить значение указанного integer-свойства
SetInteger	Установить значение указанного integer-свойства
GetDouble	Получить значение указанного double-свойства
SetDouble	Установить значение указанного double-свойства
GetString	Получить значение указанного string-свойства
SetString	Установить значение указанного string-свойства
SetSymbolPeriod	Установить значения символа и периода указанного графика
ApplyTemplate	Применяет к графику указанный шаблон
ScreenShot	Обеспечивает скриншот указанного графика
WindowOnDropped	Получить номер подокна графика, на которое брошен мышкой данный эксперт или скрипт
PriceOnDropped	Получить ценовую координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт
TimeOnDropped	Получить временную координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт
XOnDropped	Получить координату по оси X, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт
YOnDropped	Получить координату по оси Y, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт
Ввод/вывод	
virtual Save	Виртуальный метод записи в файл

© 2000-2025, MetaQuotes Ltd.

Доступ к защищенным данным	
virtual Load	Виртуальный метод чтения из файла
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

© 2000-2025, MetaQuotes Ltd.

ChartID

Получает идентификатор графика

```
long  ChartID() const
```

Возвращаемое значение

Идентификатор "привязанного" к экземпляру класса графика. Если нет "привязанного" графика возвращается -1.

© 2000-2025, MetaQuotes Ltd.

Mode (метод Get)

Получает вид графика (свечи, бары или линия).

```
ENUM_CHART_MODE  Mode() const
```

Возвращаемое значение

Вид графика (свечи, бары или линия, привязанного к экземпляру класса. Если нет "привязанного" графика возвращается WRONG_VALUE.

Mode (метод Set)

Устанавливает вид графика (свечи, бары или линия).

```
bool  Mode(
   ENUM_CHART_MODE  mode       // вид графика
   )
```

Параметры

mode

[in] Вид графика (свечи, бары или линия) из перечисления ENUM_CHART_MODE.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить вид.

© 2000-2025, MetaQuotes Ltd.

Foreground (метод Get)

Получает значение флага "Ценовой график на заднем плане".

```
bool  Foreground() const
```

Возвращаемое значение

Значение флага "Ценовой график на заднем плане" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

Foreground (метод Set)

Устанавливает значение флага "Ценовой график на заднем плане".

```
bool  Foreground(
   bool  foreground      // значение флага
   )
```

Параметры

foreground

[in] Новое значение флага "Ценовой график на заднем плане".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

Shift (метод Get)

Получает значение флага «Режим отступа ценового графика от правого края».

```
bool  Shift() const
```

Возвращаемое значение

Значение флага «Режим отступа ценового графика от правого края» графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

Shift (метод Set)

Устанавливает значение флага «Режим отступа ценового графика от правого края».

```
bool  Shift(
   bool  shift      // значение флага
   )
```

Параметры

shift

[in] Новое значение флага «Режим отступа ценового графика от правого края».

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShiftSize (метод Get)

Получает размер отступа нулевого бара от правого края в процентах.

```
double  ShiftSize() const
```

Возвращаемое значение

Размер отступа нулевого бара от правого края в процентах графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается EMPTY_VALUE.

ShiftSize (метод Set)

Устанавливает размер отступа нулевого бара от правого края в процентах.

```
bool  ShiftSize(
   double  shift_size      // значение свойства
   )
```

Параметры

shift_size

[in] Новое значение размера отступа нулевого бара от правого края в процентах.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

AutoScroll (метод Get)

Получает значение флага "Режим автоматического перехода к правому краю".

```
bool  AutoScroll() const
```

Возвращаемое значение

Значение флага "Режим автоматического перехода к правому краю" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

AutoScroll (метод Set)

Устанавливает значение флага "Режим автоматического перехода к правому краю".

```
bool  AutoScroll(
   bool  autoscroll      // значение флага
   )
```

Параметры

autoscroll

[in] Новое значение флага "Режим автоматического перехода к правому краю".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

Scale (метод Get)

Получает значение свойства "Масштаб".

```
int  Scale() const
```

Возвращаемое значение

Значение свойства "Масштаб" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается 0.

Scale (метод Set)

Устанавливает значение свойства "Масштаб".

```
bool  Scale(
   int  scale      // значение свойства
   )
```

Параметры

scale

[in] Новое значение свойства "Масштаб".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

ScaleFix (метод Get)

Получает значение флага "Режим фиксированного масштаба".

```
bool  ScaleFix() const
```

Возвращаемое значение

Значение флага "Режим фиксированного масштаба" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ScaleFix (метод Set)

Устанавливает значение флага "Режим фиксированного масштаба".

```
bool  ScaleFix(
   bool  scale_fix      // значение флага
   )
```

Параметры

scale_fix

[in] Новое значение флага "Режим фиксированного масштаба".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ScaleFix_11 (метод Get)

Получает значение флага "Режим масштаба 1:1".

```
bool  ScaleFix_11() const
```

Возвращаемое значение

Значение флага "Режим масштаба 1:1" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ScaleFix_11 (метод Set)

Устанавливает значение флага «Режим масштаба 1:1».

```
bool  ScaleFix_11(
   string  scale_11      // значение флага
   )
```

Параметры

scale_11

[in] Новое значение флага "Режим масштаба 1:1".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

FixedMax (метод Get)

Получает фиксированный максимум графика.

```
double  FixedMax() const
```

Возвращаемое значение

Фиксированный максимум графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается EMPTY_VALUE.

FixedMax (метод Set)

Устанавливает фиксированный максимум графика.

```
bool  FixedMax(
   double  max       // максимум
   )
```

Параметры

max

[in] Новый фиксированный максимум графика.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить фиксированный максимум графика.

© 2000-2025, MetaQuotes Ltd.

FixedMin (метод Get)

Получает фиксированный минимум графика.

```
double  FixedMin() const
```

Возвращаемое значение

Фиксированный минимум графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается EMPTY_VALUE.

FixedMin (метод Set)

Устанавливает фиксированный минимум графика.

```
bool  FixedMin(
   double  min       // минимум
   )
```

Параметры

min

[in] Новый фиксированный минимум графика.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить фиксированный минимум графика.

© 2000-2025, MetaQuotes Ltd.

ScalePPB (метод Get)

Получает значение флага "Режим указания масштаба в пунктах на бар".

```
bool  ScalePPB() const
```

Возвращаемое значение

Значение флага "Режим указания масштаба в пунктах на бар" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ScalePPB (метод Set)

Устанавливает значение флага "Режим указания масштаба в пунктах на бар".

```
bool  ScalePPB(
   bool  scale_ppb      // значение флага
   )
```

Параметры

scale_ppb

[in] Новое значение флага "Режим указания масштаба в пунктах на бар".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

PointsPerBar (метод Get)

Получает масштаб в пунктах на бар.

```
double  PointsPerBar() const
```

Возвращаемое значение

Масштаб в пунктах на бар графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается EMPTY_VALUE.

PointsPerBar (метод Set)

Устанавливает масштаб в пунктах на бар.

```
bool  PointsPerBar(
   double  ppb       // масштаб
   )
```

Параметры

ppb

[in] Новый масштаб в пунктах на бар.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить масштаб.

© 2000-2025, MetaQuotes Ltd.

ShowOHLC (метод Get)

Получает значение флага "Отображение значений OHLC".

```
bool  ShowOHLC() const
```

Возвращаемое значение

Значение флага "Отображение значений OHLC" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ShowOHLC (метод Set)

Устанавливает значение флага "Отображение значений OHLC".

```
bool  ShowOHLC(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение значений OHLC".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShowLineBid (метод Get)

Получает значение флага "Отображение значения Bid горизонтальной линией".

```
bool  ShowLineBid() const
```

Возвращаемое значение

Значение флага "Отображение значения Bid горизонтальной линией" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ShowLineBid (метод Set)

Устанавливает значение флага "Отображение значения Bid горизонтальной линией".

```
bool  ShowLineBid(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение значения Bid горизонтальной линией".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShowLineAsk (метод Get)

Получает значение флага "Отображение значения Ask горизонтальной линией".

```
bool  ShowLineAsk() const
```

Возвращаемое значение

Значение флага "Отображение значения Ask горизонтальной линией" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ShowLineAsk (метод Set)

Устанавливает значение флага "Отображение значения Ask горизонтальной линией".

```
bool  ShowLineAsk(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение значения Ask горизонтальной линией".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShowLastLine (метод Get)

Получает значение флага "Отображение значения Last горизонтальной линией".

```
bool  ShowLastLine() const
```

Возвращаемое значение

Значение флага "Отображение значения Last горизонтальной линией" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ShowLastLine (метод Set)

Устанавливает значение флага "Отображение значения Last горизонтальной линией".

```
bool  ShowLastLine(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение значения Last горизонтальной линией".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShowPeriodSep (метод Get)

Получает значение флага "Отображение вертикальных разделителей между соседними периодами".

```
bool  ShowPeriodSep() const
```

Возвращаемое значение

Значение флага "Отображение вертикальных разделителей между соседними периодами" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ShowPeriodSep (метод Set)

Устанавливает значение флага "Отображение вертикальных разделителей между соседними периодами".

```
bool  ShowPeriodSep(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение вертикальных разделителей между соседними периодами".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShowGrid (метод Get)

Получает значение флага "Отображение сетки".

```
bool  ShowGrid() const
```

Возвращаемое значение

Значение флага "Отображение сетки" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ShowGrid (метод Set)

Устанавливает значение флага "Отображение сетки".

```
bool  ShowGrid(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение сетки".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShowVolumes (метод Get)

Получает значение флага "Отображение объемов".

```
bool  ShowVolumes()
```

Возвращаемое значение

Значение флага "Отображение объемов" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ShowVolumes (метод Set)

Устанавливает значение флага "Отображение объемов".

```
bool  ShowVolumes(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение объемов".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShowObjectDescr (метод Get)

Получает значение флага "Отображение всплывающих описаний графических объектов".

```
bool  ShowObjectDescr() const
```

Возвращаемое значение

Значение флага "Отображение всплывающих описаний графических объектов" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

ShowObjectDescr (метод Get)

Устанавливает значение флага "Отображение всплывающих описаний графических объектов".

```
bool  ShowObjectDescr(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение всплывающих описаний графических объектов".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShowDateScale

Устанавливает значение флага "Отображение шкалы времени".

```
bool  ShowDateScale(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение шкалы времени".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ShowPriceScale

Устанавливает значение флага "Отображение ценовой шкалы".

```
bool  ShowPriceScale(
   bool  show      // значение флага
   )
```

Параметры

show

[in] Новое значение флага "Отображение ценовой шкалы".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

ColorBackground (метод Get)

Получает цвет фона.

```
color  ColorBackground() const
```

Возвращаемое значение

Значение цвет фона графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorBackground (метод Set)

Устанавливает цвет фона.

```
bool  ColorBackground(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет фона.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorForeground (метод Get)

Получает цвет осей, шкалы и строки OHLC.

```
color  ColorForeground() const
```

Возвращаемое значение

Цвет осей, шкалы и строки OHLC графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorForeground (метод Set)

Устанавливает цвет осей, шкалы и строки OHLC.

```
bool  ColorForeground(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет осей, шкалы и строки OHLC.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorGrid (метод Get)

Получаетцвет сетки.

```
color  ColorGrid() const
```

Возвращаемое значение

Цвет сетки графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorGrid (метод Set)

Устанавливает цвет сетки.

```
bool  ColorGrid(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет сетки.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorBarUp (метод Get)

Получает цвет бара вверх, тени и окантовки тела бычьей свечи.

```
color  ColorBarUp() const
```

Возвращаемое значение

Цвет бара вверх, тени и окантовки тела бычьей свечи графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorBarUp (метод Set)

Устанавливает цвет бара вверх, тени и окантовки тела бычьей свечи.

```
bool  ColorBarUp(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет бара вверх, тени и окантовки тела бычьей свечи.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorBarDown (метод Get)

Получает цвет бара вниз, тени и окантовки тела медвежьей свечи.

```
color  ColorBarDown() const
```

Возвращаемое значение

Цвет бара вниз, тени и окантовки тела медвежьей свечи графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorBarDown (метод Set)

Устанавливает цвет бара вниз, тени и окантовки тела медвежьей свечи.

```
bool  ColorBarDown(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет бара вниз, тени и окантовки тела медвежьей свечи.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorCandleBull (метод Get)

Получает цвет тела бычьей свечи.

```
color  ColorCandleBull() const
```

Возвращаемое значение

Цвет тела бычьей свечи графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorCandleBull (метод Set)

Устанавливает цвет тела бычьей свечи.

```
bool  ColorCandleBull(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет тела бычьей свечи.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorCandleBear (метод Get)

Получает цвет тела медвежьей свечи.

```
color  ColorCandleBear() const
```

Возвращаемое значение

Цвет тела медвежьей свечи графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorCandleBear (метод Set)

Устанавливает цвет тела медвежьей свечи.

```
bool  ColorCandleBear(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет тела медвежьей свечи.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorChartLine (метод Get)

Получает цвет линии графика и японских свечей "Доджи".

```
color  ColorChartLine() const
```

Возвращаемое значение

Цвет линии графика и японских свечей "Доджи" графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorChartLine (метод Set)

Устанавливает цвет линии графика и японских свечей "Доджи".

```
bool  ColorChartLine(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет линии графика и японских свечей "Доджи".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorVolumes (метод Get)

Получает цвет объемов и уровней открытия позиций.

```
color  ColorVolumes() const
```

Возвращаемое значение

Цвет объемов и уровней открытия позиций графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorVolumes (метод Set)

Устанавливает цвет объемов и уровней открытия позиций.

```
bool  ColorVolumes(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет объемов и уровней открытия позиций.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorLineBid (метод Get)

Получает цвет линии Bid-цены.

```
color  ColorLineBid() const
```

Возвращаемое значение

Цвет линии Bid-цены графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorLineBid (метод Set)

Устанавливает цвет линии Bid-цены.

```
bool  ColorLineBid(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет линии Bid-цены.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorLineAsk (метод Get)

Получает цвет линии Ask-цены.

```
color  ColorLineAsk() const
```

Возвращаемое значение

Цвет линии Ask-цены графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorLineAsk (метод Set)

Устанавливает цвет линии Ask-цены.

```
bool  ColorLineAsk(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет линии Ask-цены.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorLineLast (метод Get)

Получает цвет линии цены последней совершенной сделки.

```
color  ColorLineLast() const
```

Возвращаемое значение

Цвет линии цены последней совершенной сделки графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorLineLast (метод Set)

Устанавливает цвет линии цены последней совершенной сделки.

```
bool  ColorLineLast(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет линии цены последней совершенной сделки.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

ColorStopLevels (метод Get)

Получает цвет уровней стоп-ордеров (Stop Loss и Take Profit).

```
color  ColorStopLevels() const
```

Возвращаемое значение

Цвет уровней стоп-ордеров (Stop Loss и Take Profit) графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается CLR_NONE.

ColorStopLevels (метод Set)

Устанавливает цвет уровней стоп-ордеров (Stop Loss и Take Profit).

```
bool  ColorStopLevels(
   color  new_color      // цвет
   )
```

Параметры

new_color

[in] Новый цвет уровней стоп-ордеров (Stop Loss и Take Profit).

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить цвет.

© 2000-2025, MetaQuotes Ltd.

VisibleBars

Получает количество баров на графике, доступных для отображения.

```
int  VisibleBars() const
```

Возвращаемое значение

Количество баров, доступных для отображения на графике, привязанном к экземпляру класса. Если нет привязанного графика, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

WindowsTotal

Получает общее количество окон графика, включая подокна индикаторов.

```
int  WindowsTotal() const
```

Возвращаемое значение

Общее количество окон, включая подокна индикаторов графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

WindowIsVisible

Получает видимость подокон.

```
bool  WindowIsVisible(
   int  num      // подокно
   ) const
```

Параметры

num

[in] Номер подокна (0-основное окно).

Возвращаемое значение

Видимость подокон графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается false.

© 2000-2025, MetaQuotes Ltd.

WindowHandle

Получает хэндл графика (HWND).

```
int  WindowHandle() const
```

Возвращаемое значение

Хэндл (HWND) графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается INVALID_HANDLE.

© 2000-2025, MetaQuotes Ltd.

FirstVisibleBar

Получает номер первого видимого бара на графике.

```
int  FirstVisibleBar() const
```

Возвращаемое значение

Номер первого видимого бара графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается -1.

© 2000-2025, MetaQuotes Ltd.

WidthInBars

Получает ширину графика в барах.

```
int  WidthInBars() const
```

Возвращаемое значение

Ширину в барах графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

WidthInPixels

Получает ширину графика в пикселях.

```
int  WidthInPixels() const
```

Возвращаемое значение

Ширину в пикселях графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

HeightInPixels

Получает высоту окна в пикселях.

```
int  HeightInPixels(
   int  num      // подокно
   ) const
```

Параметры

num

[in] Номер проверяемого подокна (0-основное окно).

Возвращаемое значение

Высоту окна в пикселях графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

PriceMin

Получает минимум окна.

```
double  PriceMin(
   int  num      // подокно
   ) const
```

Параметры

num

[in] Номер подокна (0-основное окно).

Возвращаемое значение

Минимум окна графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

PriceMax

Получает максимум окна.

```
double  PriceMax(
   int  num      // подокно
   ) const
```

Параметры

num

[in] Номер подокна (0-основное окно).

Возвращаемое значение

Максимум окна графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Attach

Привязывает текущий график к экземпляру класса.

```
void  Attach()
```

Attach

Привязывает указанный график к экземпляру класса.

```
void  Attach(
   long  chart       // идентификатор графика
   )
```

Параметры

chart

[in] Идентификатор привязываемого графика.

© 2000-2025, MetaQuotes Ltd.

FirstChart

Привязывает первый график клиентского терминала к экземпляру класса.

```
void  FirstChart()
```

© 2000-2025, MetaQuotes Ltd.

NextChart

Привязывает следующий за уже привязанным, график к экземпляру класса.

```
void  NextChart()
```

© 2000-2025, MetaQuotes Ltd.

Open

Открывает график с указанными параметрами и привязывает его к экземпляру класса.

```
long  Open(
   const string      symbol_name,      // инструмент
   ENUM_TIMEFRAMES   timeframe         // период
   )
```

Параметры

symbol_name

[in] Символ графика. NULL означает символ текущего графика (к которому прикреплен эксперт).

timeframe

[in] Таймфрейм графика (из перечисления ENUM_TIMEFRAMES). 0 означает период текущего графика.

Возвращаемое значение

идентификатор графика

© 2000-2025, MetaQuotes Ltd.

Detach

Отвязывает график от экземпляра класса.

```
void  Detach()
```

© 2000-2025, MetaQuotes Ltd.

Close

Закрывает график привязанный к экземпляру класса.

```
void  Close()
```

© 2000-2025, MetaQuotes Ltd.

BringToTop

Показывает график поверх всех других.

```
bool  BringToTop() const
```

Возвращаемое значение

true в случае успеха, false в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

EventObjectCreate

Устанавливает флаг отправки сообщений о событиях создания графического объекта.

```
bool  EventObjectCreate(
   bool  flag      // флаг
   )
```

Параметры

flag

[in] Новое значение флага отправки сообщений о событиях создания графических объектов.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

EventObjectDelete

Устанавливает флаг отправки сообщений о событиях удаления графических объектов.

```
bool  EventObjectDelete(
   bool  flag      // флаг
   )
```

Параметры

flag

[in] Новое значение флага отправки сообщений о событиях удаления графических объектов.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

© 2000-2025, MetaQuotes Ltd.

IndicatorAdd

Добавляет на указанное окно графика индикатор с указанным хэндлом.

```
bool  IndicatorAdd(
   int   sub_win          // номер подокна
   int   handle           // хэндл индикатора
   );
```

Параметры

sub_win

[in] Номер подокна графика. 0 означает главное окно графика. Если указан номер несуществующего окна, то будет создано новое окно.

handle

[in] Хэндл индикатора.

Возвращаемое значение

Возвращает true в случае успеха, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Смотри также

IndicatorDelete(), IndicatorsTotal(), IndicatorName().

© 2000-2025, MetaQuotes Ltd.

IndicatorDelete

Удаляет с указанного окна графика индикатор с указанным именем.

```
bool  IndicatorDelete(
   int            sub_win      // номер подокна
   const string   name         // короткое имя индикатора
   );
```

Параметры

sub_win

[in] Номер подокна графика. 0 означает главное окно графика.

const name

[in] Короткое имя индикатора, которое задается в свойстве INDICATOR_SHORTNAME функцией IndicatorSetString(). Получить короткое имя индикатора можно функцией IndicatorName().

Возвращаемое значение

Возвращает true в случае успешного удаления индикатора, иначе false. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Если в указанном подокне графика существует несколько индикаторов с одинаковым коротким именем, то будет удален первый по порядку.

Если на значениях удаляемого индикатора построены другие индикаторы на этом же графике, то они также будут удалены.

Не следует путать короткое имя индикатора и имя файла, которое указывается при создании индикатора функциями iCustom() и IndicatorCreate(). Если короткое наименование индикатора не задается явным образом, то при компиляции в нем указывается имя файла, содержащего исходный код индикатора.

Удаление индикатора с графика не означает, что расчетная часть индикатора также будет удалена из памяти терминала. Для освобождения хэндла индикатора используйте функцию IndicatorRelease().

Необходимо правильно формировать короткое имя индикатора, которое с помощью функции IndicatorSetString() записывается в свойство INDICATOR_SHORTNAME. Мы рекомендуем, чтобы короткое имя содержало значения входных параметров индикатора, так как идентификация удаляемого с графика индикатора в функции IndicatorDelete() производится именно по короткому имени.

Смотри также

IndicatorAdd(), IndicatorsTotal(), IndicatorName(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2025, MetaQuotes Ltd.

IndicatorsTotal

Возвращает количество всех индикаторов, присоединенных к указанному окну графика.

```
int  IndicatorsTotal(
   long  chart_id,       // идентификатор графика
   int   sub_window      // номер подокна
   );
```

Параметры

chart_id

[in] Идентификатор графика. 0 означает текущий график.

sub_window

[in] Номер подокна графика. 0 означает главное окно графика.

Возвращаемое значение

Количество индикаторов на указанном окне графика. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Функция предназначена для организации перебора всех индикаторов, присоединенных к данному графику. Количества всех окон графика можно получить из свойства CHART_WINDOWS_TOTAL функцией GetInteger().

Смотри также

IndicatorAd(), IndicatorDelete(), IndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2025, MetaQuotes Ltd.

IndicatorName

Возвращает короткое имя индикатора по номеру в списке индикаторов на указанном окна графика.

```
string  IndicatorName(
   int    sub_win      // номер подокна
   int    index        // индекс индикатора в списке индикаторов, добавленных к данному
   );
```

Параметры

sub_win

[in] Номер подокна графика. 0 означает главное окно графика.

index

[in] Индекс индикатора с списке индикаторов. Нумерация индикаторов начинается с нуля, то есть самый первый индикатор в списке имеет нулевой индекс. Количество индикаторов в списке можно получить функцией IndicatorsTotal().

Возвращаемое значение

Короткое имя индикатора, которое задается в свойстве INDICATOR_SHORTNAME функцией SetString(). Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Примечание

Не следует путать короткое имя индикатора и имя файла, которое указывается при создании индикатора функциями iCustom() и IndicatorCreate(). Если короткое наименование индикатора не задается явным образом, то при компиляции в нем указывается имя файла с исходным кодом индикатора.

Необходимо правильно формировать короткое имя индикатора, которое с помощью функции IndicatorSetString() записывается в свойство INDICATOR_SHORTNAME. Мы рекомендуем, чтобы короткое имя содержало значения входных параметров индикатора, так как идентификация удаляемого с графика индикатора в функции IndicatorDelete() производится именно по короткому имени.

Смотри также

IndicatorAdd(), IndicatorDelete, IndicatorsTotal, iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2025, MetaQuotes Ltd.

Navigate

Сдвигает график.

```
bool  Navigate(
   ENUM_CHART_POSITION  position,      // позиция
   int                  shift=0       // сдвиг
   )
```

Параметры

position

[in] Позиция графика (из перечисления ENUM_CHART_POSITION), относительно которой будет произведено смещение.

shift=0

[in] Количество баров, на которое необходимо сместить график.

Возвращаемое значение

true - в случае удачи, false - если не удалось сдвинуть график.

© 2000-2025, MetaQuotes Ltd.

Symbol

Получает имя символа графика.

```
string  Symbol() const
```

Возвращаемое значение

Имя символа графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается "".

© 2000-2025, MetaQuotes Ltd.

Period

Получает значение периода графика.

```
ENUM_TIMEFRAMES  Period() const
```

Возвращаемое значение

Период графика (из перечисления ENUM_TIMEFRAMES), привязанного к экземпляру класса. Если нет привязанного графика, возвращается 0.

© 2000-2025, MetaQuotes Ltd.

Redraw

Перерисовывает график, привязанный к экземпляру класса.

```
void  Redraw()
```

© 2000-2025, MetaQuotes Ltd.

GetInteger

Возвращает значение соответствующего свойства указанного графика. Свойство графика должно быть типа integer. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
long  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // идентификатор свойства
   int                          sub_window=0     // номер подокна
   ) const
```

2. Помещает значение свойства в приемную переменную, передаваемую по ссылке последним параметром, в случае успеха.

```
bool  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,        // идентификатор свойства
   int                          sub_window,     // номер подокна
   long&                        value           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор свойства графика (из перечисления ENUM_CHART_PROPERTY_INTEGER).

sub_window

[in] Номер подокна графика.

value

[in] Ссылка на переменную, принимающую значение запрашиваемого свойства.

Возвращаемое значение

Значение свойства графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается -1.

Для второго варианта вызова возвращает true, если данное свойство поддерживается и значение было помещено в переменную value, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

SetInteger

Устанавливает значение integer-свойства графика.

```
bool  SetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,      // идентификатор свойства
   long                         value         // значение
   )
```

Параметры

prop_id

[in] Идентификатор свойства графика (из перечисления ENUM_CHART_PROPERTY_INTEGER).

value

[in] Новое значение свойства.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить integer-свойство.

© 2000-2025, MetaQuotes Ltd.

GetDouble

Возвращает значение соответствующего свойства указанного графика. Свойство графика должно быть типа double. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
double  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // идентификатор свойства
   int                         sub_window=0     // номер подокна
   ) const
```

2. Помещает значение свойства в приемную переменную, передаваемую по ссылке последним параметром, в случае успеха.

```
bool  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,        // идентификатор свойства
   int                         sub_window,     // номер подокна
   double&                     value           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор свойства графика (из перечисления ENUM_CHART_PROPERTY_DOUBLE).

sub_window

[in] Номер подокна графика.

value

[in] Ссылка на переменную, принимающую значение запрашиваемого свойства.

Возвращаемое значение

Значение свойства графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается EMPTY_VALUE.

Для второго варианта вызова возвращает true - в случае удачи, false - если не удалось получить значение свойства. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

SetDouble

Устанавливает значение double-свойства графика.

```
bool  SetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,      // идентификатор свойства
   double                      value         // значение
   )
```

Параметры

prop_id

[in] Идентификатор свойства графика (из перечисления ENUM_CHART_PROPERTY_DOUBLE).

value

[in] Новое значение свойства.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить double-свойство.

© 2000-2025, MetaQuotes Ltd.

GetString

Возвращает значение соответствующего свойства указанного графика. Свойство графика должно быть типа string. Существует 2 варианта функции.

1. Непосредственно возвращает значение свойства.

```
string  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id      // идентификатор свойства
   ) const
```

2. Помещает значение свойства в приемную переменную, передаваемую по ссылке последним параметром, в случае успеха.

```
bool  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,     // идентификатор свойства
   string&                     value        // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор свойства графика (из перечисления ENUM_CHART_PROPERTY_STRING).

value

[in] Ссылка на переменную, принимающую значение запрашиваемого свойства.

Возвращаемое значение

Значение свойства графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается "".

Для второго варианта вызова возвращает true, если данное свойство поддерживается и значение было помещено в переменную value, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2025, MetaQuotes Ltd.

SetString

Устанавливает значение string-свойства графика.

```
bool  SetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,      // идентификатор свойства
   string                      value         // значение
   )
```

Параметры

prop_id

[in] Идентификатор свойства графика (из перечисления ENUM_CHART_PROPERTY_STRING).

value

[in] Новое значение свойства.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить string-свойство.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Изменяет символ и период графика, привязанного к экземпляру класса.

```
bool  SetSymbolPeriod(
   const string      symbol_name,      // инструмент
   ENUM_TIMEFRAMES   timeframe         // период
   )
```

Параметры

symbol_name

[in] Новый символ графика. NULL означает символ текущего графика (к которому прикреплен эксперт).

timeframe

[in] Новый период графика (из перечисления ENUM_TIMEFRAMES). 0 означает период текущего графика.

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойства.

© 2000-2025, MetaQuotes Ltd.

ApplyTemplate

Применяет к графику указанный шаблон.

```
bool  ApplyTemplate(
   const string  filename      // шаблон
   )
```

Параметры

filename

[in] Имя файла, содержащего шаблон.

Возвращаемое значение

true - в случае удачи, false - если не удалось применить шаблон.

© 2000-2025, MetaQuotes Ltd.

ScreenShot

Обеспечивает скриншот указанного графика в его текущем состоянии в формате gif.

```
bool  ScreenShot(
   string           filename,                  // имя файла
   int              width,                     // ширина
   int              height,                    // высота
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // тип выравнивания
   ) const
```

Параметры

filename

[in] Имя файла скриншота.

width

[in] Ширина скриншота в пикселях.

height

[in] Высота скриншота в пикселях.

align_mode=ALIGN_RIGHT

[in] Режим вывода узкого скриншота.

Возвращаемое значение

true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

WindowOnDropped

Получает номер подокна графика, на которое брошен мышкой данный эксперт или скрипт.

```
int  WindowOnDropped() const
```

Возвращаемое значение

Номер подокна графика, на которое брошен мышкой данный эксперт или скрипт. 0 означает главное окно графика.

© 2000-2025, MetaQuotes Ltd.

PriceOnDropped

Получает ценовую координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

```
double  PriceOnDropped() const
```

Возвращаемое значение

Ценовая координата, соответствующая точке, в которой брошен мышкой данный эксперт или скрипт.

© 2000-2025, MetaQuotes Ltd.

TimeOnDropped

Получает временную координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

```
datetime  TimeOnDropped() const
```

Возвращаемое значение

Временная координата, соответствующая точке, в которой брошен мышкой данный эксперт или скрипт.

© 2000-2025, MetaQuotes Ltd.

XOnDropped

Получает координату по оси X, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

```
int  XOnDropped() const
```

Возвращаемое значение

Координата по оси X, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

© 2000-2025, MetaQuotes Ltd.

YOnDropped

Получает координату по оси Y, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

```
int  YOnDropped() const
```

Возвращаемое значение

Координата по оси Y, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт.

© 2000-2025, MetaQuotes Ltd.

Save

Сохраняет данные элемента списка в файле.

```
virtual bool  Save(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Load

Загружает параметры объекта из файла.

```
virtual bool  Load(
   int  file_handle      // хэндл файла
   )
```

Параметры

file_handle

[in] хэндл ранее открытого, при помощи функции FileOpen(...), бинарного файла

Возвращаемое значение

true - в случае успешного завершения, false - в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

Type

Получает идентификатор типа.

```
virtual int  Type() const
```

Возвращаемое значение

Идентификатор типа (для CChart - 0x1111).

© 2000-2025, MetaQuotes Ltd.

Научные графики

Графическая библиотека, содержащая классы и глобальные функции для быстрой отрисовки пользовательских графиков на чарте. Библиотека предоставляет удобные готовые решения для построения осей, кривых, а также методы быстрого доступа к изменению общих свойств пользовательского графика.

Для того чтобы начать работать с библиотекой, достаточно прочитать статью Визуализируй это! Графическая библиотека в MQL5 как аналог plot из R.

Графическая библиотека размещается в рабочем каталоге терминала в папке Include\Graphics.

Класс	Описание
CAxis	Класс для работы с координатными осями
ColorGenerator	Класс, задающий цветовую схему по умолчанию
CCurve	Класс для работы с кривыми
CGraphic	Базовый класс для создания пользовательских графиков

© 2000-2025, MetaQuotes Ltd.

GraphPlot

Функции быстрой отрисовки кривых.

Версия для отрисовки одной кривой по координатам Y.

```
string  GraphPlot(
   const double      &y[],                  // координаты Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // тип кривой
   )
```

Примечание

Координатами X для данной кривой будут служить индексы массива Y.

Версия для отрисовки одной кривой по паре координат X и Y

```
string  GraphPlot(
   const double      &x[],                  // координаты X
   const double      &y[],                  // координаты Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // тип кривой
   )
```

Версия для отрисовки двух кривых по паре координат X и Y

```
string  GraphPlot(
   const double      &x1[],                 // координаты X
   const double      &y1[],                 // координаты Y
   const double      &x2[],                 // координаты X
   const double      &y2[],                 // координаты Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // тип кривой
   )
```

Версия для отрисовки трех кривых по паре координат X и Y

```
string  GraphPlot(
   const double      &x1[],                 // координаты X
   const double      &y1[],                 // координаты Y
   const double      &x2[],                 // координаты X
   const double      &y2[],                 // координаты Y
   const double      &x3[],                 // координаты X
   const double      &y3[],                 // координаты Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // тип кривой
   )
```

© 2000-2025, MetaQuotes Ltd.

Версия для отрисовки кривой по координатам точек CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points[],            // координаты кривой
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // тип кривой
   )
```

Версия для отрисовки двух кривых по координатам точек CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points1[],           // координаты кривой
   const CPoint2D   &points2[],           // координаты кривой
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // тип кривой
   )
```

Версия для отрисовки трех кривых по координатам точек CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points1[],           // координаты кривой
   const CPoint2D   &points2[],           // координаты кривой
   const CPoint2D   &points3[],           // координаты кривой
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // тип кривой
   )
```

Версия для отрисовки кривой по указателю на функцию CurveFunction

```
string  GraphPlot(
   CurveFunction    function,             // указатель на функцию
   const double     from,                 // начальное значение аргумента
   const double     to,                   // конечное значение аргумента
   const double     step,                 // приращение по аргументу
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // тип кривой
   )
```

Версия для отрисовки двух кривых по указателям на функции CurveFunction

```
string  GraphPlot(
   CurveFunction    function1,            // указатель на функцию
   CurveFunction    function2,            // указатель на функцию
   const double     from,                 // начальное значение аргумента
   const double     to,                   // конечное значение аргумента
   const double     step,                 // приращение по аргументу
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // тип кривой
   )
```

© 2000-2025, MetaQuotes Ltd.

Версия для отрисовки трех кривых по указателям на функции CurveFunction

```
string  GraphPlot(
   CurveFunction    function1,             // указатель на функцию
   CurveFunction    function2,             // указатель на функцию
   CurveFunction    function3,             // указатель на функцию
   const double     from,                  // начальное значение аргумента
   const double     to,                    // конечное значение аргумента
   const double     step,                  // приращение по аргументу
   ENUM_CURVE_TYPE  type=CURVE_POINTS      // тип кривой
   )
```

Параметры

&x[]

[in] Координаты X.

&y[]

[in] Координаты Y.

&x1[]

[in] Координаты X для первой кривой.

&y1[]

[in] Координаты X для второй кривой.

&x2[]

[in] Координаты X для второй кривой.

&y2[]

[in] Координаты Y для второй кривой.

&x3[]

[in] Координаты X для третьей кривой.

&y3[]

[in] Координаты Y для третьей кривой.

&points[]

[in] Координаты точек кривой.

&points1[]

[in] Координаты точек первой кривой.

&points2[]

[in] Координаты точек второй кривой.

&points3[]

[in] Координаты точек третьей кривой.

© 2000-2025, MetaQuotes Ltd.

function

[in] Указатель на функцию CurveFunction.

function1

[in] Указатель на первую функцию.

function2

[in] Указатель на вторую функцию.

function3

[in] Указатель на третью функцию.

from

[in] Соответствует первой координате X.

to

[in] Соответствует последней координате X.

step

[in] Параметр для расчета координат X.

type=CURVE_POINTS

[in] Тип кривой.

Возвращаемое значение

Имя графического ресурса.

© 2000-2025, MetaQuotes Ltd.

CAxis

Класс CAxis — вспомогательный класс графической библиотеки для работы с координатными осями.

Описание

Класс CAxis обеспечивает хранение и получение различных параметров координатных осей. В классе реализована возможность динамического автомасштабирования координатной оси.

Декларация

```
class CAxis
```

Заголовок

```
#include <Graphics\Axis.mqh>
```

Методы класса

Метод	Описание
AutoScale	Получить/установить флаг автомасштабирования
Min	Получить/установить минимальное значение оси
Max	Получить/установить максимальное значение оси
Step	Возвращает значение шага по оси
Name	Получить/установить имя оси
Color	Получить/установить цвет оси
ValuesSize	Получить/установить размер цифр на оси
ValuesWidth	Получить/установить максимальную отображаемую длину цифр на оси
ValuesFormat	Получить/установить формат цифр на оси
ValuesDateTimeMode	Возвращает формат преобразования даты в строку.
ValuesFunctionFormat	Возвращает указатель на функцию, определяющую формат вывода значений на оси.
ValuesFunctionFormatCBData	Возвращает указатель на объект, в котором может содержаться дополнительная информация для преобразования значений оси.
NameSize	Получить/установить размер шрифта для имени оси
ZeroLever	Получить/установить значение "нулевого рычага"

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
DefaultStep	Получить/установить начальное значение шага по оси
MaxLabels	Получить/установить максимальное количество цифр на оси
MinGrace	Получить/установить значение "допуска" для минимума оси
MaxGrace	Получить/установить значение "допуска" для максимума оси
SelectAxisScale	Выполняет автомасштабирование оси.

© 2000-2025, MetaQuotes Ltd.

AutoScale (метод Get)

Возвращает флаг, отмечающий, нужно ли выполнять автомасштабирование.

```
bool  AutoScale()
```

Возвращаемое значение

Значение флага.

Примечание

true — выполнять автомасштабирование.

false — не выполнять автомасштабирование.

AutoScale (метод Set)

Устанавливает флаг, отмечающий, нужно ли выполнять автомасштабирование.

```
void  AutoScale(
   const bool  auto      // значение флага
   )
```

Параметры

auto

[in]

Примечание

true — выполнять автомасштабирование.

false — не выполнять автомасштабирование.

© 2000-2025, MetaQuotes Ltd.

Min (метод Get)

Возвращает минимальное значение оси.

```
double  Min()
```

Возвращаемое значение

Минимальное значение оси.

Min (метод Set)

Устанавливает минимальное значение оси.

```
void  Min(
   const double  min       // минимальное значение
   )
```

Параметры

min

[in] Минимальное значение.

© 2000-2025, MetaQuotes Ltd.

Max (метод Get)

Возвращает максимальное значение оси.

```
double  Max()
```

Возвращаемое значение

Максимальное значение оси.

Max (метод Set)

Устанавливает максимальное значение оси графика.

```
void  Max(
   const double  max       // максимальное значение
   )
```

Параметры

max

[in] Максимальное значение оси графика.

© 2000-2025, MetaQuotes Ltd.

Step (метод Get)

Возвращает значение шага по оси.

```
double  Step()
```

Возвращаемое значение

Значение шага.

© 2000-2025, MetaQuotes Ltd.

Name (метод Get)

Возвращает имя оси.

```
string  Name()
```

Возвращаемое значение

Имя оси.

Name (метод Set)

Устанавливает имя оси.

```
void  Name(
   const string  name      // имя оси
   )
```

Параметры

name

[in] Имя оси.

© 2000-2025, MetaQuotes Ltd.

Color (метод Get)

Возвращает цвет оси.

```
color  Color()
```

Возвращаемое значение

Цвет оси.

Color (метод Set)

Устанавливает цвет оси.

```
void  Color(
   const color  clr      // цвет оси
   )
```

Параметры

clr

[in] Цвет оси.

© 2000-2025, MetaQuotes Ltd.

ValuesSize (метод Get)

Возвращает размер цифр на оси.

```
int  ValuesSize()
```

Возвращаемое значение

Размер цифр, отображаемых на оси.

ValuesSize (метод Set)

Устанавливает размер цифр на оси.

```
void  ValuesSize(
   const int  size      // размер цифр на оси
   )
```

Параметры

size

[in] Размер цифр на оси

© 2000-2025, MetaQuotes Ltd.

ValuesWidth (метод Get)

Возвращает максимально допустимую длину в пикселях для отображения цифр на оси.

```
int  ValuesWidth()
```

Возвращаемое значение

Длина отображения цифр на оси в пикселях.

Примечание

Если длина в пикселях для задаваемой цифры превышает максимально допустимую длину отображения, то она будет обрезаться и оканчиваться многоточием.

ValuesWidth (метод Set)

Устанавливает максимально допустимую длину в пикселях для отображения цифр на оси.

```
void  ValuesWidth(
   const int  width      // максимально допустимая длина в пикселях
   )
```

Параметры

width

[in] Максимально допустимая длина отображения цифр на оси.

Примечание

Если длина в пикселях для задаваемой цифры превышает максимально допустимую длину отображения, то она будет обрезаться и оканчиваться многоточием.

© 2000-2025, MetaQuotes Ltd.

ValuesFormat (метод Get)

Возвращает формат отображения цифр на оси.

```
string  ValuesFormat()
```

Возвращаемое значение

Формат отображения цифр.

ValuesFormat (метод Set)

Устанавливает формат отображения цифр на оси.

```
void  ValuesFormat(
   const string  format      // формат отображения цифр на оси
   )
```

Параметры

format

[in] Формат отображения цифр на оси.

© 2000-2025, MetaQuotes Ltd.

ValuesDateTimeMode (Метод Get)

Возвращает формат преобразования даты в строку.

```
int  ValuesDateTimeMode()
```

Возвращаемое значение

Формат преобразования даты в строку.

ValuesDateTimeMode (Метод Set)

Устанавливает формат преобразования даты в строку.

```
void  ValuesDateTimeMode(
   const int  mode      // формат преобразования даты в строку
   )
```

Параметры

mode

[in] Формат преобразования.

Примечание

Более подробно о форматах преобразования даты в строку смотрите в описании функции TimeToString().

© 2000-2025, MetaQuotes Ltd.

ValuesFunctionFormat (Метод Get)

Возвращает указатель на функцию, определяющую формат вывода значений на оси.

```
DoubleToStringFunction  ValuesFunctionFormat()
```

Возвращаемое значение

Указатель на функцию, определяющую формат вывода значений на оси.

ValuesFunctionFormat (Метод Set)

Устанавливает указатель на функцию, которая определяет формат вывода значений на оси.

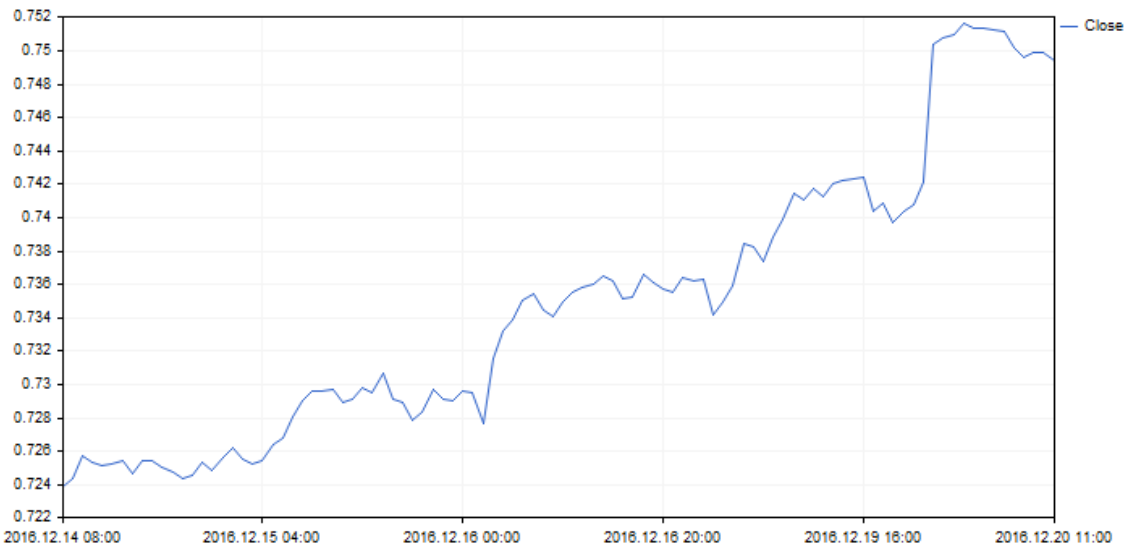
```
void  ValuesFunctionFormat(
   DoubleToStringFunction  func      // функция для конвертирования числовых значений
   )
```

Параметры

func

[in] Пользовательская функция для конвертирования числовых значений в строку.

Пример:



Формат вывода значений на оси X был изменен с помощью следующего кода:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                DateAxisGraphic.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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();
  }
```

© 2000-2025, MetaQuotes Ltd.

ValuesFunctionFormatCBData (Метод Get)

Возвращает указатель на объект, в котором может содержаться дополнительная информация для преобразования значений оси.

```
void*  ValuesFunctionFormatCBData()
```

Возвращаемое значение

Указатель на объект, в котором может содержаться дополнительная информация для преобразования значений оси.

ValuesFunctionFormatCBData (Метод Set)

Устанавливает указатель на объект класса, в котором может содержаться дополнительная информация для преобразования значений оси.

```
void  ValuesFunctionFormatCBData(
   void*  cbdata      // указатель на объект класса
   )
```

Параметры

cbdata

[in] Указатель на объект любого класса, содержащий дополнительную информацию для преобразования значений оси

© 2000-2025, MetaQuotes Ltd.

NameSize (метод Get)

Возвращает размер шрифта для имени оси.

```
int  NameSize()
```

Возвращаемое значение

Размер шрифта, которым задается имя оси.

NameSize (метод Set)

Устанавливает размер шрифта для имени оси.

```
void  NameSize(
   const int  size      // размер шрифта для имени оси
   )
```

Параметры

size

[in] Размер шрифта, которым задается имя оси.

© 2000-2025, MetaQuotes Ltd.

ZeroLever (метод Get)

Возвращает значение "нулевого рычага".

```
double  ZeroLever()
```

Возвращаемое значение

"Нулевой рычаг".

Примечание

Это значение используется для определения того, когда диапазон шкалы оси должен быть расширен, чтобы включить нулевое значение.

ZeroLever (метод Set)

Устанавливает значение "нулевого рычага".

```
void  ZeroLever(
   const double  value      // значение "нулевого рычага"
   )
```

Параметры

value

[in] Значение "нулевого рычага".

Примечание

Это значение используется для определения того, когда диапазон шкалы оси должен быть расширен, чтобы включить нулевое значение.

© 2000-2025, MetaQuotes Ltd.

DefaultStep (метод Get)

Возвращает начальное значение шага по оси.

```
double  DefaultStep()
```

Возвращаемое значение

Шаг по оси.

DefaultStep (метод Set)

Устанавливает начальное значение шага по оси.

```
void  DefaultStep(
   const double  value      // шаг по оси
   )
```

Параметры

value

[in] Начальное значение шага по оси.

© 2000-2025, MetaQuotes Ltd.

MaxLabels (метод Get)

Возвращает максимальное допустимое количество отображаемых цифр на оси.

```
double  MaxLabels()
```

Возвращаемое значение

Максимальное количество цифр на оси.

MaxLabels (метод Set)

Устанавливает максимальное допустимое количество отображаемых цифр на оси.

```
void  MaxLabels(
   const double  value      // максимальное количество
   )
```

Параметры

value

[in] Максимально допустимое количество отображаемых на оси цифр

© 2000-2025, MetaQuotes Ltd.

MinGrace (метод Get)

Возвращает значение "допуска", применяемое к минимуму оси.

```
double  MinGrace()
```

Возвращаемое значение

Значение "допуска" для минимума оси.

Примечание

Это значение выражается в виде части общей длины оси. Например, пусть значения оси находятся в пределах от 4.0 до 16.0, тогда ее длина будет равна 12.0. Если MinGrace равен 0.1, то 10% от длины оси (или 1.2) будет вычтено из значения минимума. В результате ось будет покрывать интервал от 2.8 до 16.0.

MinGrace (метод Set)

Устанавливает значение "допуска", применяемое к минимуму оси.

```
void  MinGrace(
   const double  value      // значение "допуска"
   )
```

Параметры

value

[in] Значение "допуска", применяемое к минимуму оси.

Примечание

Это значение выражается в виде части общей длины оси. Например, пусть значения оси находятся в пределах от 4.0 до 16.0, тогда ее длина будет равна 12.0. Если MinGrace равен 0.1, то 10% от длины оси (или 1.2) будет вычтено из значения минимума. В результате ось будет покрывать интервал от 2.8 до 16.0.

© 2000-2025, MetaQuotes Ltd.

MaxGrace (метод Get)

Возвращает значение "допуска", применяемое к максимуму оси.

```
double  MaxGrace()
```

Возвращаемое значение

Значение "допуска" для максимума оси.

Примечание

Это значение выражается в виде части общей длины оси. Например, пусть значения оси находятся в пределах от 4.0 до 16.0, тогда ее длина будет равна 12.0. Если MaxGrace равен 0.1, то 10% от длины оси (или 1.2) будет добавлено к значению максимума. В результате ось будет покрывать интервал от 4.0 до 17.2.

MaxGrace (метод Set)

Устанавливает значение "допуска", применяемое к максимуму оси.

```
void  MaxGrace(
   const double  value      // значение "допуска"
   )
```

Параметры

value

[in] Значение допуска, применямое к максимуму оси.

Примечание

Это значение выражается в виде части общей длины оси. Например, пусть значения оси находятся в пределах от 4.0 до 16.0, тогда ее длина будет равна 12.0. Если MinGrace равен 0.1, то 10% от длины оси (или 1.2) будет вычтено из значения минимума. В результате, ось будет покрывать интервал от 2.8 до 16.0.

© 2000-2025, MetaQuotes Ltd.

SelectAxisScale

Выполняет автомасштабирование оси.

```
void  SelectAxisScale()
```

© 2000-2025, MetaQuotes Ltd.

CColorGenerator

Класс CColorGenerator — вспомогательный класс графической библиотеки для работы с цветовой палитрой.

Описание

Класс CColorGenerator содержит начальную палитру цветов, используемых по умолчанию для кривых (если цвет не указан пользователем).

Если все цвета из начальной палитры уже были использованы, то производится автоматическая генерация новых цветов и перезаполнение палитры.

Декларация

```
class CColorGenerator
```

Заголовок

```
#include <Graphics\ColorGenerator.mqh>
```

Методы класса

Метод	Описание
Next	Возвращает следующий цвет из палитры
Reset	Приводит генератор к первоначальному состоянию

© 2000-2025, MetaQuotes Ltd.

Next

Возвращает следующий цвет из палитры.

```
uint  Next()
```

Возвращаемое значение

Цвет.

Примечание

Если все цвета из палитры были перебраны, то начинается автоматическая генерация новых цветов, которыми будут замещены в палитре старые.

© 2000-2025, MetaQuotes Ltd.

Reset

Приводит генератор к первоначальному состоянию.

```
void  Reset()
```

© 2000-2025, MetaQuotes Ltd.

CCurve

Класс CCurve является классом для работы со свойствами кривых, создаваемых на графике.

Описание

Класс CCurve обеспечивает установку, хранение, получение координат и различных свойств кривых при работе с классом CGraphic.

Предусмотрены три режима отрисовки кривых: точками, линиями и гистограммой. Для каждого режима отрисовки в классе реализованы отдельные параметры.

Декларация

```
class CCurve : public CObject
```

Заголовок

```
#include <Graphics\Curve.mqh>
```

Иерархия наследования

CObject

CCurve

Методы класса

Метод	Описание
Type	Возвращает тип кривой
Name	Возвращает имя кривой
Color	Возвращает цвет кривой
XMax	Возвращает максимальное значение функции по оси X
XMin	Возвращает минимальное значение функции по оси X
YMax	Возвращает максимальное значение функции по оси Y
YMin	Возвращает минимальное значение функции по оси Y
Size	Возвращает количество точек, которые определяют кривую
PointsSize	Получить/установить линейные размеры точек, определяющих кривую
PointsFill	Получить/установить флаг заливки точек, определяющих кривую

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
PointsColor	Получить/установить цвет заливки точек
GetX	Получает в массив значения X всех точек кривой
GetY	Получает в массив значения Y всех точек кривой
LinesStyle	Получить/установить стиль линии при отрисовке кривой линиями
LinesIsSmooth	Получить/установить флаг сглаживания при отрисовке линиями
LinesSmoothTension	Получить/установить параметр сглаживания кривой при отрисовке линиями
LinesSmoothStep	Получить/установить длину аппроксимирующих линий для сглаживания при отрисовке линиями
LinesWidth	Получить/установить ширину линий при отрисовке кривой линиями.
HistogramWidth	Получить/установить ширину столбцов при отрисовке гистограммой
CustomPlotCBData	Получить/установить указатель на объект, который будет использоваться при пользовательском режиме отрисовки кривой.
CustomPlotFunction	Получить/установить указатель на функцию, которая реализует пользовательский режим отрисовки кривой.
PointsType	Получить/установить флаг, указывающий на тип точек, использующихся при отрисовке кривой точками.
StepsDimension	Получить/установить значение, указывающее на измерение, по которому делается шаг при ступенчатой отрисовке кривой.
TrendLineCoefficients	Получить/установить коэффициенты трендовой линии для записи их в массив.
TrendLineColor	Получить/установить цвет трендовой линии для кривой.
TrendLineVisible	Получить/установить флаг видимости трендовой линии.
Update	Обновляет координаты кривой.
Visible	Получить/установить флаг, указывающий, будет ли видна функция на графике.

Методы унаследованные от CObject

© 2000-2025, MetaQuotes Ltd.

Prev, Prev, Next, Next, Save, Load, Compare

© 2000-2025, MetaQuotes Ltd.

Type

Возвращает тип кривой.

```
ENUM_CURVE_TYPE  Type()
```

Возвращаемое значение

Тип кривой.

© 2000-2025, MetaQuotes Ltd.

Name

Возвращает имя кривой.

```
string  Name()
```

Возвращаемое значение

Имя кривой.

© 2000-2025, MetaQuotes Ltd.

Color

Возвращает цвет кривой.

```
uint  Color()
```

Возвращаемое значение

Цвет кривой.

© 2000-2025, MetaQuotes Ltd.

XMax

Возвращает максимальное значение для функции по оси X (только действительные числа).

```
double  XMax()
```

Возвращаемое значение

Максимальное действительное число среди всех аргументов для данной функции.

© 2000-2025, MetaQuotes Ltd.

XMin

Возвращает минимальное значение для функции по оси X (только действительные значения).

```
double  XMin()
```

Возвращаемое значение

Минимальное действительное число среди всех аргументов для данной функции.

© 2000-2025, MetaQuotes Ltd.

YMax

Возвращает максимальное значение для функции по оси Y (только действительные числа).

```
double  YMax()
```

Возвращаемое значение

Максимальное значение для данной функции по оси Y (только действительные числа).

© 2000-2025, MetaQuotes Ltd.

YMin

Возвращает минимальное значение для функции по оси Y (только действительные числа).

```
double  YMin()
```

Возвращаемое значение

Минимальное значение для данной функции по оси Y (только действительные числа).

© 2000-2025, MetaQuotes Ltd.

Size

Возвращает количество точек, которые определяют данную кривую.

```
int  Size()
```

Возвращаемое значение

Количество точек, по которым определена кривая.

© 2000-2025, MetaQuotes Ltd.

PointsSize (метод Get)

Возвращает линейные размеры точек, определяющих данную кривую при ее точечной отрисовке, в пикселях.

```
int  PointsSize()
```

Возвращаемое значение

Размер определяющих кривую точек в пикселях.

PointsSize (метод Set)

Устанавливает размер точек, определяющих данную кривую при ее точечной отрисовке, в пикселях.

```
void  PointsSize(
   const int  size      // размер точек в пикселях
   )
```

Параметры

size

[in] Размер точек, определяющих данную кривую, в пикселях.

© 2000-2025, MetaQuotes Ltd.

PointsFill (метод Get)

Возвращает флаг, указывающий, нужно ли выполнять заливку для точек, определяющих кривую при отрисовке точками.

```
bool  PointsFill ()
```

Возвращаемое значение

Значение флага.

Примечание

true — выполнять заливку

false — не выполнять заливку

PointsFill (метод Set)

Устанавливает флаг, указывающий, нужно ли выполнять заливку для точек, определяющих кривую при отрисовке точками.

```
void  PointsFill(
   const bool  fill      // значение флага
   )
```

Параметры

fill

[in] Значение флага.

Примечание

true — выполнять заливку

false — не выполнять заливку

© 2000-2025, MetaQuotes Ltd.

PointsColor (метод Get)

Возвращает цвет заливки точек.

```
uint  PointsColor ()
```

Возвращаемое значение

Цвет заливки точек, определяющих данную кривую.

PointsColor (метод Set)

Устанавливает цвет заливки точек

```
void  PointsColor(
   const uint  clr      //цвет заливки точек
   )
```

Параметры

clr

[in] Цвет заливки точек, определяющих данную кривую.

© 2000-2025, MetaQuotes Ltd.

GetX

Получает в массив значения X всех точек кривой.

```
void  GetX(
   double&  x[]      // массив для записи значений X
   )
```

Параметры

x[]

[out] Массив для получения значений X всех точек кривой.

Примечание

Каждая точка кривой задается парой значений X и Y. Эти значения не являются координатами в пикселях для отрисовки в классе CGraphic.

© 2000-2025, MetaQuotes Ltd.

GetY

Получает в массив значения Y всех точек кривой.

```
void  GetY(
   double&  y[]      // массив для записи значений Y
   )
```

Параметры

y[]

[out] Массив для получения значений Y всех точек кривой.

Примечание

Каждая точка кривой задается парой значений X и Y. Эти значения не являются координатами в пикселях для отрисовки в классе CGraphic.

© 2000-2025, MetaQuotes Ltd.

LinesStyle (метод Get)

Возвращает стиль линии при отрисовке кривой линиями.

```
ENUM_LINE_STYLE  LinesStyle()
```

Возвращаемое значение

Стиль линии.

LinesStyle (метод Set)

Устанавливает стиль линий при отрисовке кривой линиями.

```
void  LinesStyle (
   ENUM_LINE_STYLE  style      // стиль линии
   )
```

Параметры

style

[in] Стиль линии.

© 2000-2025, MetaQuotes Ltd.

LinesIsSmooth (метод Get)

Возвращает флаг, определяющий, нужно ли выполнять сглаживание при отрисовке кривой линиями.

```
bool  LinesIsSmooth()
```

Возвращаемое значение

Значение флага

Примечание

true — выполнять сглаживание

false — не выполнять сглаживание

LinesIsSmooth (метод Set)

Устанавливает флаг, определяющий, нужно ли выполнять сглаживание при отрисовке кривой линиями.

```
void  LinesIsSmooth(
   const bool  smooth      // значение флага
   )
```

Параметры

smooth

[in] Значение флага

Примечание

true — выполнять сглаживание

false — не выполнять сглаживание

© 2000-2025, MetaQuotes Ltd.

LinesSmoothTension (метод Get)

Возвращает параметр сглаживания кривой при отрисовке линиями.

```
double  LinesSmoothTension()
```

Возвращаемое значение

Значение параметра сглаживания

Примечание

Значение tension находится в диапазоне (0.0; 1.0].

LinesSmoothTension (метод Set)

Устанавливает параметр сглаживания кривой при отрисовке линиями.

```
void  LinesSmoothTension(
   const double  tension      // значение параметра
   )
```

Параметры

tension

[in] Значение параметра сглаживания.

Примечание

Значение tension находится в диапазоне (0.0; 1.0].

© 2000-2025, MetaQuotes Ltd.

LinesSmoothStep (метод Get)

Возвращает длину аппроксимирующих линий для сглаживания при отрисовке линиями.

```
double  LinesSmoothStep()
```

Возвращаемое значение

Длина аппроксимирующих линий в пикселях.

LinesSmoothStep (метод Set)

Устанавливает длину аппроксимирующих линий для сглаживания при отрисовке линиями.

```
void  LinesSmoothStep(
   const double  step      // длина линий
   )
```

Параметры

step

[in] Длина аппроксимирующих линий

© 2000-2025, MetaQuotes Ltd.

LinesEndStyle (Метод Set)

Возвращает флаг, указывающий на стиль отрисовки концов линий при отрисовке кривой линиями.

```
ENUM_LINE_END  LinesEndStyle()
```

Возвращаемое значение

Значение флага, указывающего на стиль концов линий при отрисовке кривой линиями.

LinesEndStyle (Метод Get)

Устанавливает флаг, указывающий на стиль отрисовки концов линий при отрисовке кривой линиями.

```
void  LinesEndStyle(
   ENUM_LINE_END  end_style      // значение флага
   )
```

Параметры

end_style

[in] Значение флага, указывающего на стиль концов линии при отрисовке кривой линиями.

© 2000-2025, MetaQuotes Ltd.

LinesWidth (Метод Get)

Возвращает ширину линий при отрисовке кривой линиями.

```
int  LinesWidth()
```

Возвращаемое значение

Ширина линий.

LinesWidth (Метод Set)

Устанавливает ширину линий при отрисовке кривой линиями.

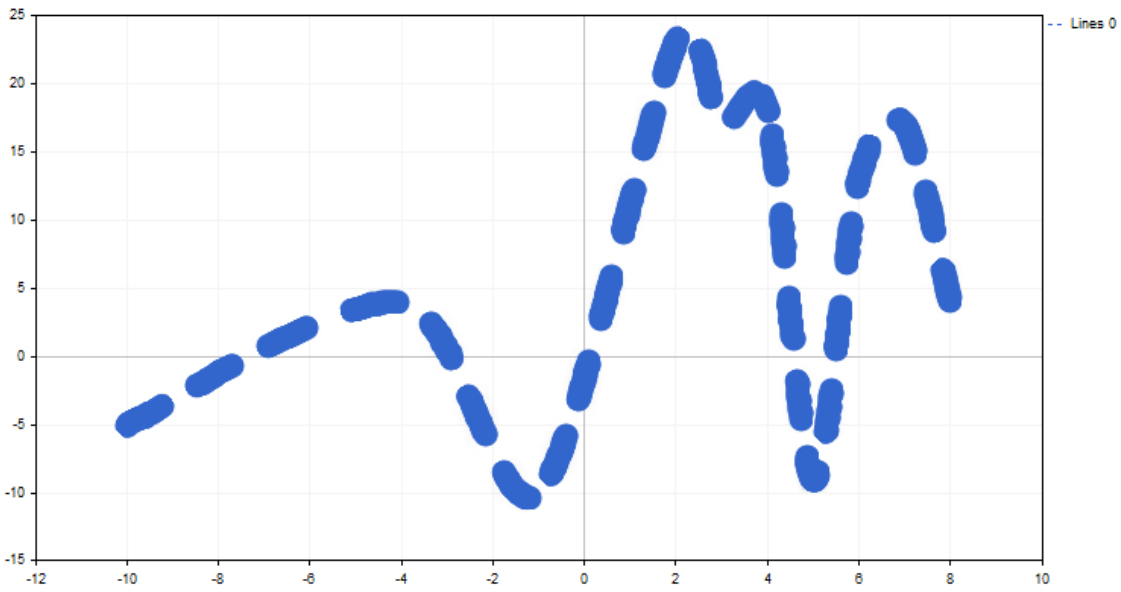
```
void  LinesWidth(
   const int  width      // ширина линий
   )
```

Параметры

width

[in] Ширина линий при отрисовке кривой линиями.

Пример:



Толщина линии изменена с помощью следующего кода:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             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();
  }
```

© 2000-2025, MetaQuotes Ltd.

HistogramWidth (метод Get)

Возвращает ширину столбцов при отрисовке гистограммой.

```
int  HistogramWidth()
```

Возвращаемое значение

Ширина столбцов в пикселях.

HistogramWidth (метод Set)

Устанавливает ширину столбцов при отрисовке гистограммой.

```
void  HistogramWidth(
   const int  width      // ширина столбцов
   )
```

Параметры

width

[in] Ширина столбцов в пикселях.

© 2000-2025, MetaQuotes Ltd.

CustomPlotCBData (Метод Get)

Возвращает указатель на объект, который будет использоваться при пользовательском режиме отрисовки кривой.

```
void*  CustomPlotCBData()
```

Возвращаемое значение

Указатель на объект для пользовательского режима отрисовки кривой.

CustomPlotCBData (Метод Set)

Устанавливает указатель на объект класса, который будет использоваться при пользовательском режиме отрисовки кривой.

```
void  CustomPlotCBData(
   void*  cbdata      // указатель на объект
   )
```

Параметры

cbdata

[in] Указатель на объект любого класса, который будет использоваться при пользовательском режиме отрисовки кривой

© 2000-2025, MetaQuotes Ltd.

CustomPlotFunction (Метод Get)

Возвращает указатель на функцию, которая реализует пользовательский режим отрисовки кривой.

```
PlotFucntion  CustomPlotFunction()
```

Возвращаемое значение

Указатель на функцию, реализующую пользовательский режим отрисовки кривой.

CustomPlotFunction (Метод Set)

Устанавливает указатель на функцию, которая реализует пользовательский режим отрисовки кривой.

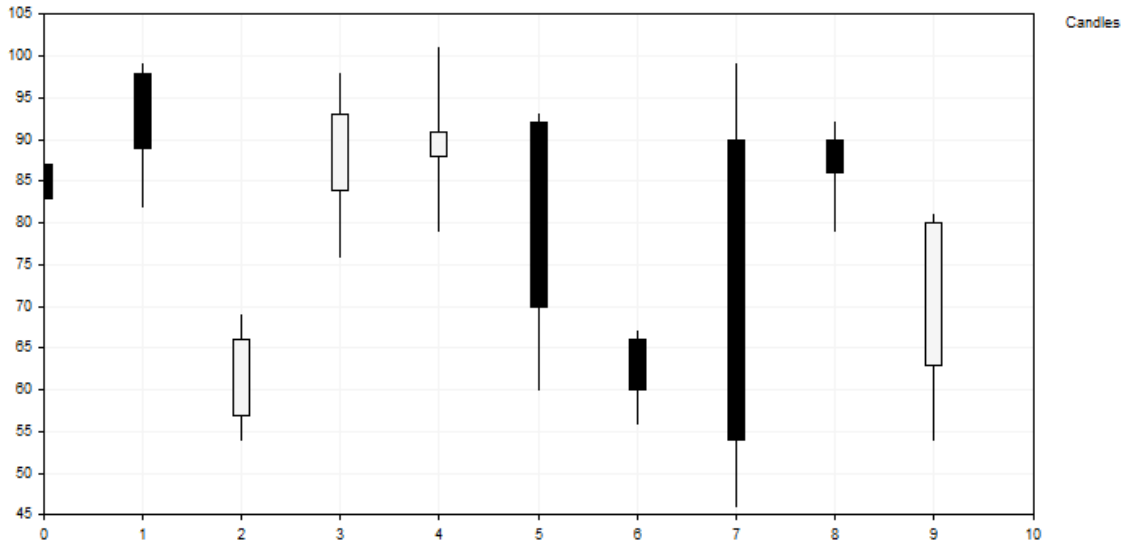
```
void  CustomPlotFunction(
   PlotFucntion  func      // указатель на функцию
   )
```

Параметры

func

[in] Указатель на функцию, реализующую пользовательский режим отрисовки кривой

Пример:



Эта кривая из баров была построена с помощью следующего кода:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              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));
```

© 2000-2025, MetaQuotes Ltd.

```
      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.CandleColorDecr
      //--- 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));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- sets the graphic properties
   graphic.YAxis().Max((int)max);
   graphic.YAxis().Min((int)min);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

PointsType (Метод Get)

Возвращает флаг, указывающий на тип точек, использующихся при отрисовке кривой точками.

```
ENUM_POINT_TYPE  PointsType()
```

Возвращаемое значение

Значение флага, указывающего на тип точек.

PointsType (Метод Set)

Устанавливает флаг, указывающий на тип точек, использующихся при отрисовке кривой точками.

```
void  PointsType(
   ENUM_POINT_TYPE  type      // значение флага
   )
```

Параметры

type

[in] Значение флага, указывающего на тип точек, использующихся при отрисовке кривой точками.

© 2000-2025, MetaQuotes Ltd.

StepsDimension (Метод Get)

Возвращает значение, указывающее на измерение, по которому делается шаг при ступенчатой отрисовке кривой.

```
int  StepsDimension()
```

Возвращаемое значение

Измерение, по которому делается шаг при ступенчатой отрисовке кривой.

StepsDimension (Метод Set)

Уставливает значение, указывающее на измерение, по которому делается шаг при ступенчатой отрисовке кривой.

```
void  StepsDimension(
   const int  dimension      // измерение
   )
```

Параметры

dimension

[in] Измерение (может быть либо 0, либо 1).

Примечание

0 — x (сначала рисуется горизонтальная линия, затем вертикальная).

1 — y (сначала рисуется вертикальная линия, затем горизонтальная).

© 2000-2025, MetaQuotes Ltd.

TrendLineCoefficients (Метод Get)

Возвращает коэффициенты трендовой линии для записи их в массив.

```
double&  TrendLineCoefficients()
```

Возвращаемое значение

Коэффициенты трендовой линии.

TrendLineCoefficients (Метод Set)

Получает коэффициенты трендовой линии для записи их в массив.

```
void  TrendLineCoefficients(
   double&  coefficients[]      // массив для записи коэффициентов
   )
```

Параметры

coefficients[]

[out] Массив для записи коэффициентов.

© 2000-2025, MetaQuotes Ltd.

TrendLineColor (Метод Get)

Возвращает цвет трендовой линии для кривой.

```
uint  TrendLineColor()
```

Возвращаемое значение

Цвет трендовой линии.

TrendLineColor (Метод Set)

Устанавливает цвет трендовой линии для кривой.

```
void  TrendLineColor(
   const uint  clr      // цвет трендовой линии
   )
```

Параметры

clr

[in] Цвет линии

© 2000-2025, MetaQuotes Ltd.

TrendLineVisible (Метод Get)

Возвращает флаг, который определяет видимость трендовой линии.

```
bool  TrendLineVisible()
```

Возвращаемое значение

Значение флага, который указывает, видна ли трендовая линия.

TrendLineVisible (Метод Set)

Устанавливает флаг, который определяет видимость трендовой линии.

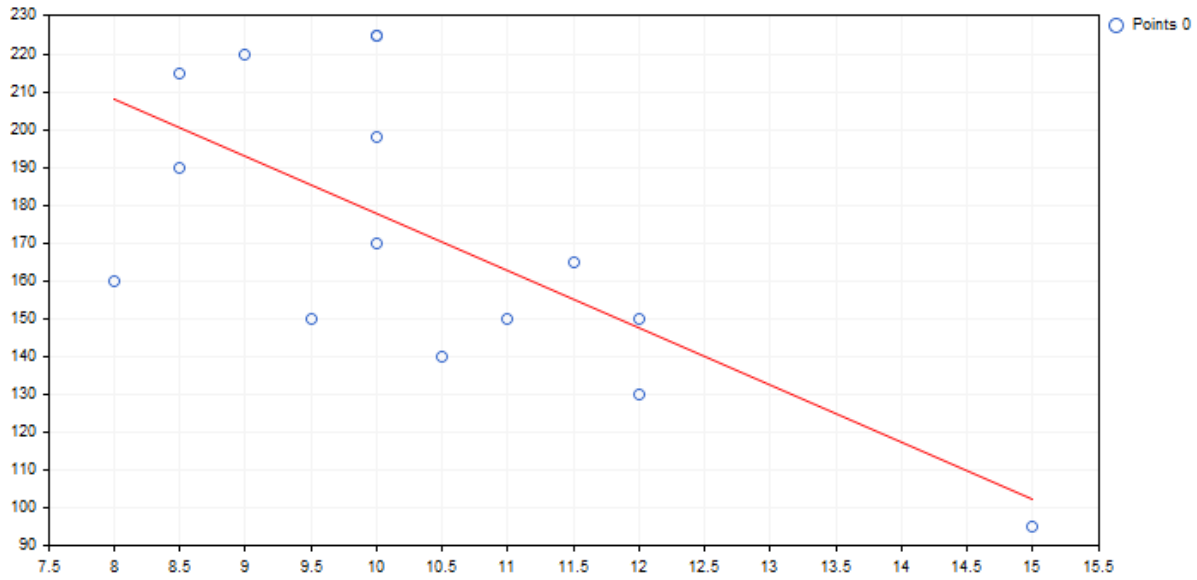
```
void  TrendLineVisible(
   const bool  visible      // значение флага
   )
```

Параметры

visible

[in] Значение флага, определяющего видимость трендовой линии.

Пример:



Код построения вышеприведенной трендовой линии и ее отображения на графике:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              TrendLineGraphic.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              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();
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

Обновляет координаты кривой.

Версия для работы по координате Y. Координатами X в данном случае будут индексы переданного массива.

```
void  Update(
   const double&  y[]   // Координаты Y
   )
```

Версия для работы по паре координат X и Y.

```
void  Update(
   const double&  x[],     // Координаты X
   const double&  y[]      // Координаты Y
   )
```

Версия для работы с точками CPoint2D.

```
void  Update(
   const CPoint2D&  points[]      // Координаты кривой
   )
```

Версия для работы с указателем на функцию CurveFunction.

```
void  Update(
   CurveFunction  function,     // указатель на функцию, описывающую кривую
   const double   from,         // начальное значение аргумента функции
   const double   to,           // конечное значение аргумента функции
   const double   step          // приращение аргумента
   )
```

Параметры

x[]

[in] Координаты X.

y[]

[in] Координаты Y.

points[]

[in] Координаты кривой.

function

[in] Указатель на функцию, описывающую кривую

from

[in] Начальное значение аргумента функции

to

[in] Конечное значение аргумента функции

step

© 2000-2025, MetaQuotes Ltd.

[in] Приращение аргумента

© 2000-2025, MetaQuotes Ltd.

Visible (Метод Get)

Возвращает флаг, указывающий, будет ли видна функция на графике.

```
void  Visible(
   const bool  visible      //
   )
```

Возвращаемое значение

Значение флага видимости функции на графике.

Visible (Метод Set)

Устанавливает флаг, указывающий, будет ли видна функция на графике.

```
void  Visible(
   const bool  visible      // значение флага
   )
```

Параметры

visible

[in] Значение флага видимости функции на графике.

© 2000-2025, MetaQuotes Ltd.

CGraphic

Класс CGraphic — базовый класс для создания пользовательских графиков.

Описание

Класс CGraphic обеспечивает многочисленные аспекты работы с пользовательскими графиками.

В классе хранятся основные элементы графика, задаются их параметры, реализуется отрисовка.

Также данный класс хранит кривые для данного графика и обеспечивает различные варианты их отображения.

Декларация

```
class CGraphic
```

Заголовок

```
#include <Graphics\Graphic.mqh>
```

Методы класса

Метод	Описание
Create	Создает графический ресурс, привязанный к объекту чарта
Destroy	Удаляет с чарта график и уничтожает графический ресурс
Update	Отображает на экране сделанные изменения
ChartObjectName	Возвращает имя привязанного к графику объекта
ResourceName	Возвращает имя графического ресурса
XAxis	Возвращает указатель на ось X
YAxis	Возвращает указатель на ось Y
GapSize	Получить/установить размер отступов между элементами графика
BackgroundColor	Получить/установить цвет фона
BackgroundMain	Получить/установить текст заголовка графика
BackgroundMainSize	Получить/установить размер шрифта заголовка
BackgroundMainColor	Получить/установить цвет заголовка графика
BackgroundSub	Получить/установить текст подзаголовка
BackgroundSubSize	Получить/установить размер шрифта заголовка
BackgroundSubColor	Получить/установить цвет подзаголовка графика

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
GridLineColor	Получить/установить цвет линий сетки
GridBackgroundColor	Получить/установить цвет фона сетки
GridCircleRadius	Получить/установить радиус точек в узлах сетки
GridCircleColor	Получить/установить цвет точек в узлах сетки
GridHasCircle	Получить/установить флаг отрисовки точек в узлах сетки
GridAxisLineColor	Возвращает значение цвета реальных осей графика.
HistoryNameWidth	Получить/установить максимально допустимую длину для отображения имени кривой
HistoryNameSize	Получить/установить размер шрифта имени кривой
HistorySymbolSize	Получить/установить размер символов условных обозначений
TextAdd	Добавляет текст на график
LineAdd	Добавляет линию на график
CurveAdd	Создает и добавляет кривую на график
CurvePlot	Отрисовывает ранее созданную кривую по индексу
CurvePlotAll	Отрисовывает все ранее созданные кривые
CurveGetByIndex	Получает кривую по заданному индексу
CurveGetByName	Получает кривую по заданному имени
CurveRemoveByIndex	Удаляет кривую по заданному индексу.
CurveRemoveByName	Удаляет кривую по заданному имени.
CurvesTotal	Возвращает количество кривых для данного графика.
MarksToAxisAdd	Добавляет разметку шкалы на ось графика
MajorMarkSize	Получить/установить размер "засечек" шкалы на оси графика
FontSet	Установить параметры текущего шрифта
FontGet	Получить параметры текущего шрифта
Attach	Получить/создать графический ресурс и привязать его к экземпляру класса CGraphic
CalculateMaxMinValues	Выполняет расчет (перерасчет) минимальных и максимальных значений графика по обеим осям.

© 2000-2025, MetaQuotes Ltd.

Метод	Описание
Height	Возвращает высоту графика в пикселях.
IndentDown	Получить/установить отступ графика от нижней границы.
IndentLeft	Получить/установить отступ графика от левой границы.
IndentRight	Получить/установить отступ графика от правой границы.
IndentUp	Получить/установить отступ графика от верхней границы.
Redraw	Перерисовывает график.
ResetParameters	Сбрасывает параметры, необходимые для перерисовки графика.
ScaleX	Масштабирует значение по оси X.
ScaleY	Масштабирует значение по оси Y.
SetDefaultParameters	Устанавливает параметры графика в значения по умолчанию.
Width	Возвращает ширину графика в пикселях.

© 2000-2025, MetaQuotes Ltd.

Create

Создает графический ресурс, привязанный к объекту чарта.

```
bool  Create(
   const long     chart,       // идентификатор чарта
   const string   name,        // имя
   const int      subwin,      // номер подокна
   const int      x1,          // координата x1
   const int      y1,          // координата y1
   const int      x2,          // координата x2
   const int      y2           // координата y1
   )
```

Параметры

chart

[in] Идентификатор графика.

name

[in] Имя.

subwin

[in] Номер подокна.

x1

[in] Координата X1.

y1

[in] Координата Y1.

x2

[in] Координата X2.

y2

[in] Координата Y2.

© 2000-2025, MetaQuotes Ltd.

Destroy

Удаляет с чарта график и уничтожает графический ресурс.

```
void  Destroy()
```

© 2000-2025, MetaQuotes Ltd.

Update

Отображает на экране сделанные изменения.

```
void  Update(
   const bool  redraw=true      // флаг
   )
```

Параметры

redraw=true

[in] Значение флага

© 2000-2025, MetaQuotes Ltd.

ChartObjectName

Получает имя привязанного к графику объекта.

```
string  ChartObjectName()
```

Возвращаемое значение

Имя привязанного к графику объекта.

© 2000-2025, MetaQuotes Ltd.

ResourceName

Получает имя графического ресурса.

```
string  ResourceName()
```

Возвращаемое значение

Имя графического ресурса.

© 2000-2025, MetaQuotes Ltd.

XAxis

Возвращает указатель на ось X.

```
CAxis  *XAxis()
```

Возвращаемое значение

Указатель на ось X.

© 2000-2025, MetaQuotes Ltd.

YAxis

Возвращает указатель на ось Y.

```
CAxis  *YAxis()
```

Возвращаемое значение

Указатель на ось Y.

© 2000-2025, MetaQuotes Ltd.

GapSize (метод Get)

Возвращает размер отступов между элементами графика.

```
int  GapSize()
```

Возвращаемое значение

Размер отступа в пикселях.

GapSize (метод Set)

Устанавливает размер отступов между элементами графика.

```
void  GapSize(
   const int  size     // размер отступа
   )
```

Параметры

size

[in] Размер отступа в пикселях.

© 2000-2025, MetaQuotes Ltd.

BackgroundColor (метод Get)

Возвращает цвет фона.

```
color  BackgroundColor()
```

BackgroundColor (метод Set)

Устанавливает цвет фона.

```
void  BackgroundColor(
   const color  clr      // цвет фона
   )
```

Параметры

clr

[in] Цвет фона.

© 2000-2025, MetaQuotes Ltd.

BackgroundMain (метод Get)

Возвращает текст заголовка графика

```
string  BackgroundMain()
```

BackgroundMain (метод Set)

Устанавливает текст заголовка графика.

```
void  BackgroundMain(
   const string  main      // текст заголовка
   )
```

Параметры

main

[in] Текст заголовка графика

© 2000-2025, MetaQuotes Ltd.

BackgroundMainSize (метод Get)

Возвращает размер заголовка.

```
int  BackgroundMainSize()
```

Возвращаемое значение

Размер шрифта заголовка.

BackgroundMainSize (метод Set)

Устанавливает размер заголовка.

```
void  BackgroundMainSize(
   const int  size     // размер заголовка
   )
```

Параметры

size

[in] Размер шрифта заголовка.

© 2000-2025, MetaQuotes Ltd.

BackgroundMainColor (метод Get)

Возвращает цвет заголовка.

```
color  BackgroundMainColor()
```

Возвращаемое значение

Цвет заголовка.

BackgroundMainColor (метод Set)

Устанавливает цвет заголовка.

```
void  BackgroundMainColor(
   const color  clr      // цвет заголовка
   )
```

Параметры

clr

[in] Цвет заголовка.

© 2000-2025, MetaQuotes Ltd.

BackgroundSub (метод Get)

Возвращает текст подзаголовка.

```
string  BackgroundSub()
```

Возвращаемое значение

Текст подзаголовка.

BackgroundSub (метод Set)

Устанавливает текст подзаголовка.

```
void  BackgroundSub (метод Set)(
   const string  sub       // текст подзаголовка
   )
```

Параметры

sub

[in] Текст подзаголовка.

© 2000-2025, MetaQuotes Ltd.

BackgroundSubSize (метод Get)

Возвращает размер шрифта подзаголовка

```
int BackgroundSubSize()
```

BackgroundSubSize (метод Get)

Устанавливает размер подзаголовка

```
void  BackgroundSubSize(
   const int  size      // размер шрифта подзаголовка
   )
```

Параметры

size

[in] Размер шрифта подзаголовка

© 2000-2025, MetaQuotes Ltd.

BackgroundSubColor (метод Get)

Возвращает цвет подзаголовка.

```
color  BackgroundSubColor()
```

BackgroundSubColor (метод Set)

Устанавливает цвет подзаголовка.

```
void  BackgroundSubColor(
   const color  clr      // цвет подзаголовка
   )
```

Параметры

clr

[in] Цвет подзаголовка.

© 2000-2025, MetaQuotes Ltd.

GridLineColor (метод Get)

Возвращает цвет линий сетки

```
color  GridLineColor()
```

Возвращаемое значение

Цвет линий сетки.

GridLineColor (метод Set)

Устанавливает цвет линий сетки.

```
void  GridLineColor(
   const color  clr      // цвет линий
   )
```

Параметры

clr

[in] Цвет линий сетки.

© 2000-2025, MetaQuotes Ltd.

GridBackgroundColor (метод Get)

Возвращает цвет фона сетки

```
color  GridBackgroundColor()
```

Возвращаемое значение

Цвет фона сетки

GridBackgroundColor (метод Set)

Устанавливает цвет заднего фона сетки

```
void  GridBackgroundColor(
   const color  clr      // цвет фона сетки
   )
```

Параметры

clr

[in] Цвет фона сетки

© 2000-2025, MetaQuotes Ltd.

GridCircleRadius (метод Get)

Возвращает радиус точек в узлах сетки.

```
int  GridCircleRadius()
```

Возвращаемое значение

Радиус точек в пикселях.

GridCircleRadius (метод Set)

Устанавливает радиус точек в узлах сетки

```
void  GridCircleRadius(
   const int  r      // радиус
   )
```

Параметры

r

[in] Радиус точек в пикселях.

© 2000-2025, MetaQuotes Ltd.

GridCircleColor (метод Get)

Возвращает цвет точек в узлах сетки.

```
color  GridCircleColor()
```

Возвращаемое значение

Цвет точек.

GridCircleColor (метод Set)

Устанавливает цвет точек в узлах сетки.

```
void  GridCircleColor(
   const color  clr      // цвет точек
   )
```

Параметры

clr

[in] Цвет точек.

© 2000-2025, MetaQuotes Ltd.

GridHasCircle (метод Get)

Возвращает флаг, указывающий, нужно ли отрисовывать точки в узлах сетки.

```
bool  GridHasCircle()
```

Возвращаемое значение

Значение флага.

Примечание

true — отрисовывать точки

false — не отрисовывать точки

GridHasCircle (метод Set)

Устанавливает флаг, указывающий, нужно ли отрисовывать точки в узлах сетки.

```
void  GridHasCircle(
   const bool  has
   )
```

Параметры

has

[in] Значение флага.

Примечание

true — отрисовывать точки

false — не отрисовывать точки

© 2000-2025, MetaQuotes Ltd.

GridAxisLineColor (Метод Get)

Возвращает значение цвета реальных осей графика.

```
uint  GridAxisLineColor()
```

Возвращаемое значение

Цвет реальных осей графика.

GridAxisLineColor (Метод Set)

Устанавливает значение цвета реальных осей графика.

```
void  GridAxisLineColor(
   const uint  clr      // цвет осей графика
   )
```

Параметры

clr

[in] Цвет реальных осей графика.

© 2000-2025, MetaQuotes Ltd.

HistoryNameWidth (метод Get)

Возвращает максимально допустимую длину для отображения имени кривой.

```
int  HistoryNameWidth()
```

Возвращаемое значение

Максимальная длина в пикселях.

Примечание

Если имя кривой превышает максимально допустимую длину, то оно будет обрезано, а в его конец будет добавлено многоточие.

HistoryNameWidth (метод Set)

Устанавливает максимально допустимую длину для отображения имени кривой.

```
void  HistoryNameWidth(
   const int  width      // максимальная длина
   )
```

Параметры

width

[in] Максимальная длина в пикселях.

Примечание

Если имя кривой превышает максимально допустимую длину, то оно будет обрезано, а в его конец будет добавлено многоточие.

© 2000-2025, MetaQuotes Ltd.

HistoryNameSize (метод Get)

Возвращает размер шрифта имени кривой.

```
int  HistoryNameSize()
```

Возвращаемое значение

Размер шрифта имени кривой.

HistoryNameSize (метод Set)

Устанавливает размер шрифта имени кривой.

```
void  HistoryNameSize (метод Set)(
   const int  size      // размер шрифта имени
   )
```

Параметры

size

[in] Размер шрифта имени.

© 2000-2025, MetaQuotes Ltd.

HistorySymbolSize (метод Get)

Возвращает размер символов условных обозначений к графику

```
int  HistorySymbolSize()
```

Возвращаемое значение

Размер символов условных обозначений

HistorySymbolSize (метод Set)

Устанавливает размер символов условных обозначений к графику

```
void  HistorySymbolSize(
   const int  size      // размер символа
   )
```

Параметры

size

[in] Размер символов условных обозначений.

© 2000-2025, MetaQuotes Ltd.

TextAdd

Добавляет текст на график.

Версия для работы с парой координат X и Y

```
void  TextAdd(
   const int     x,              // координата X
   const int     y,              // координата Y
   const string  text,           // текст
   const uint    clr,            // цвет
   const uint    alignment=0     // выравнивание
   )
```

Версия для CPoint

```
void  TextAdd(
   const CPoint  &point,         // координата точки
   const string  text,           // текст
   const uint    clr,            // цвет
   const uint    alignment=0     // выравнивание
   )
```

Параметры

x

[in] Координата X.

y

[in] Координата Y.

&point

[in] Координата точки.

text

[in] Текст.

clr

[in] Цвет.

alignment=0

[in] Выравнивание.

© 2000-2025, MetaQuotes Ltd.

LineAdd

Добавляет линию на график.

Версия для работы с парами координат X и Y

```
void  LineAdd(
   const int   x1,         // координата x1
   const int   y1,         // координата y1
   const int   x2,         // координата x2
   const int   y2,         // координата y2
   const uint  clr,        // цвет
   const uint  style       // стиль
   )
```

Версия для CPoint

```
void  LineAdd2(
   const CPoint  &point1,     // координата первой точки
   const CPoint  &point2,     // координата второй точки
   const uint    clr,         // цвет
   const uint    style        // стиль
   )
```

Параметры

x1

[in] Координата X1.

y1

[in] Координата Y1.

x2

[in] Координата X2.

y2

[in] Координата Y2.

&point1

[in] Координата первой точки.

&point2

[in] Координата второй точки.

clr

[in] Цвет.

style

[in] Стиль.

© 2000-2025, MetaQuotes Ltd.

CurveAdd

Создаёт и добавляет новую кривую на график.

Версия для работы по координате Y (цвет кривой задается автоматически)

```
CCurve* CurveAdd(
   const double      &y[],          // координаты Y
   ENUM_CURVE_TYPE   type,          // тип кривой
   const string      name=NULL      // имя кривой
   )
```

Примечание

Координатами X для данной кривой будут служить индексы массива Y.

Версия для работы по паре координат X и Y (цвет кривой задается автоматически)

```
CCurve*  CurveAdd(
   const double      &x[],          // координаты X
   const double      &y[],          // координаты Y
   ENUM_CURVE_TYPE   type,          // тип кривой
   const string      name=NULL      // имя кривой
   )
```

Версия для работы с точками CPoint2D (цвет кривой задается автоматически)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // координаты точек
   ENUM_CURVE_TYPE   type,          // тип кривой
   const string      name=NULL      // имя кривой
   )
```

Версия для работы с указателем на функцию CurveFunction (цвет кривой задается автоматически)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // указатель на функцию
   const double      from,          // начальное значение аргумента
   const double      to,            // конечное значение аргумента
   const double      step,          // приращение по аргументу
   ENUM_CURVE_TYPE   type,          // тип кривой
   const string      name=NULL      // имя кривой
   )
```

© 2000-2025, MetaQuotes Ltd.

Версия для работы по координате Y (цвет кривой задается пользователем)

```
CCurve*  CurveAdd(
   const double      &y[],          // координаты Y
   const uint        clr,           // цвет кривой
   ENUM_CURVE_TYPE   type,          // тип кривой
   const string      name=NULL      // имя кривой
   )
```

Примечание

Координатами X для данной кривой будут служить индексы массива Y.

Версия для работы по паре координат X и Y (цвет кривой задается пользователем)

```
CCurve*  CurveAdd(
   const double      &x[],          // координаты X
   const double      &y[],          // координаты Y
   const uint        clr,           // цвет кривой
   ENUM_CURVE_TYPE   type,          // тип кривой
   const string      name=NULL      // имя кривой
   )
```

Версия для работы с точками CPoint2D (цвет кривой задается пользователем)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // координаты точек
   const uint        clr,           // цвет кривой
   ENUM_CURVE_TYPE   type,          // тип кривой
   const string      name=NULL      // имя кривой
   )
```

Версия для работы с указателем на функцию CurveFunction (цвет кривой задается пользователем)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // указатель на функцию
   const double      from,          // начальное значение аргумента
   const double      to,            // конечное значение аргумента
   const double      step,          // приращение по аргументу
   const uint        clr,           // цвет кривой
   ENUM_CURVE_TYPE   type,          // тип кривой
   const string      name=NULL      // имя кривой
   )
```

© 2000-2025, MetaQuotes Ltd.

Параметры

&x[]

[in] Координата X.

&y[]

[in] Координата Y.

&points[]

[in] Координаты точек.

function

[in] Указатель на функцию.

from

[in] Начальное значение аргумента.

to

[in] Конечное значение аргумента.

step

[in] Приращение по аргументу.

type

[in] Тип кривой.

name=NULL

[in] Имя кривой.

clr

[in] Цвет кривой.

Возвращаемое значение

Указатель на созданную кривую.

© 2000-2025, MetaQuotes Ltd.

CurvePlot

Отображает ранее созданную кривую с заданным индексом.

```
bool  CurvePlot(
   const int  index      // индекс
   )
```

Параметры

index

[in] Индекс кривой

Возвращаемое значение

true в случае удачи, иначе false.

© 2000-2025, MetaQuotes Ltd.

CurvePlotAll

Отображает все ранее добавленные к графику кривые.

```
bool  CurvePlotAll()
```

Возвращаемое значение

true в случае удачи, иначе false.

© 2000-2025, MetaQuotes Ltd.

CurveGetByIndex

Получает кривую по заданному индексу.

```
CCurve*  CurveGetByIndex(
   const int  index      // индекс кривой
   )
```

Параметры

index

[in] Индекс кривой.

Возвращаемое значение

Указатель на кривую с заданным индексом.

© 2000-2025, MetaQuotes Ltd.

CurveGetByName

Получает кривую по заданному имени.

```
CCurve*  CurveGetByName(
   const string  name      // имя кривой
   )
```

Параметры

name

[in] Имя кривой.

Возвращаемое значение

Указатель на первую найденную кривую с заданным именем.

© 2000-2025, MetaQuotes Ltd.

CurveRemoveByIndex

Удаляет кривую по заданному индексу.

```
bool  CurveRemoveByIndex(
   const int  index      // индекс кривой
   )
```

Параметры

index

[in] Индекс кривой, которую требуется удалить.

Возвращаемое значение

true — в случае удачи, иначе — false.

© 2000-2025, MetaQuotes Ltd.

CurveRemoveByName

Удаляет кривую по заданному имени.

```
bool  CurveRemoveByName(
   const string  name      // имя кривой
   )
```

Параметры

name

[in] Имя кривой, которую требуется удалить.

Возвращаемое значение

true — в случае удачи, иначе — false.

© 2000-2025, MetaQuotes Ltd.

CurvesTotal

Возвращает количество кривых для данного графика.

```
int  CurvesTotal()
```

Возвращаемое значение

Количество кривых.

Примечание

Считаются все кривые, принадлежащие данному графику, вне зависимости от их стиля отрисовки и видимости.

© 2000-2025, MetaQuotes Ltd.

MarksToAxisAdd

Добавляет разметку шкалы ("засечки") на указанную ось графика.

```
bool  MarksToAxisAdd(
   const double         &marks[],        // координаты "засечек"
   const int            mark_size,       // размер "засечек"
   ENUM_MARK_POSITION   position,        // расположение "засечек"
   const int            dimension=0      // измерение
   )
```

Параметры

&marks[]

[in] Координаты "засечек"

mark_size

[in] Размер "засечек"

position

[in] Расположение "засечек"

dimension=0

[in] 0 — добавление на ось X,

1 — добавление на ось Y

Возвращаемое значение

true в случае удачи, иначе false.

© 2000-2025, MetaQuotes Ltd.

MajorMarkSize (метод Get)

Возвращает размер "засечек" шкалы на координатных осях.

```
int  MajorMarkSize()
```

MajorMarkSize (метод Set)

Устанавливает размер "засечек" шкалы на координатных осях.

```
void  MajorMarkSize(
   const int  size      // размер "засечек"
   )
```

Параметры

size

[in] Размер "засечек" в пикселях.

© 2000-2025, MetaQuotes Ltd.

FontSet

Устанавливает параметры текущего шрифта.

```
bool  FontSet(
   const string  name,         // имя
   const int     size,         // размер
   const uint    flags=0,      // флаги
   const uint    angle=0       // угол
   )
```

Параметры

name

[in] Имя.

size

[in] Размер.

flags=0

[in] Флаги.

angle=0

[in] Угол.

Возвращаемое значение

true — в случае удачи, иначе — false.

© 2000-2025, MetaQuotes Ltd.

FontGet

Получает параметры текущего шрифта.

```
void  FontGet(
   string  &name,       // имя
   int     &size,       // размер
   uint    &flags,      // флаги
   uint    &angle       // угол
   )
```

Параметры

&name

[out] Имя.

&size

[out] Размер.

&flags

[out] Флаги.

&angle

[out] Угол.

© 2000-2025, MetaQuotes Ltd.

Attach

Версия для получения из объекта OBJ_BITMAP_LABEL графического ресурса и привязки его к экземпляру класса CGraphic:

```
bool  Attach(
   const long     chart_id,      // идентификатор чарта
   const string   objname        // имя графического объекта
   )
```

Версия для создания графического ресурса для объекта OBJ_BITMAP_LABEL и привязки его к экземпляру класса CGraphic:

```
bool  Attach(
   const long     chart_id,      // идентификатор чарта
   const string   objname,       // имя графического объекта
   const int      width,         // ширина картинки
   const int      height         // высота картинки
   )
```

Параметры

chart_id

[in] Идентификатор чарта.

objname

[in] Наименование (имя) графического объекта.

width

[in] Ширина картинки в ресурсе.

height

[in] Высота картинки в ресурсе.

Возвращаемое значение

true — в случае удачи, false — если не удалось привязать объект.

© 2000-2025, MetaQuotes Ltd.

CalculateMaxMinValues

Выполняет расчет (перерасчет) минимальных и максимальных значений графика по обеим осям.

```
void  CalculateMaxMinValues()
```

© 2000-2025, MetaQuotes Ltd.

Height

Возвращает высоту графика в пикселях.

```
int  Height()
```

Возвращаемое значение

Высота графика в пикселях.

© 2000-2025, MetaQuotes Ltd.

IndentDown (Метод Get)

Возвращает отступ графика от нижней границы.

```
int  IndentDown()
```

Возвращаемое значение

Размер отступа в пикселях.

IndentDown (Метод Set)

Устанавливает отступ графика от нижней границы.

```
void  IndentDown(
   const int  down      // размер отступа
   )
```

Параметры

down

[in] Размер отступа в пикселях.

© 2000-2025, MetaQuotes Ltd.

IndentLeft (Метод Get)

Возвращает отступ графика от левой границы.

```
int  IndentLeft()
```

Возвращаемое значение

Размер отступа в пикселях.

IndentLeft (Метод Set)

Устанавливает отступ графика от левой границы.

```
void  IndentLeft(
   const int  left      // размер отступа
   )
```

Параметры

left

[in] Размер отступа в пикселях.

© 2000-2025, MetaQuotes Ltd.

IndentRight (Метод Get)

Возвращает отступ графика от правой границы.

```
int  IndentRight()
```

Возвращаемое значение

Размер отступа в пикселях.

IndentRight (Метод Set)

Устанавливает отступ графика от правой границы.

```
void  IndentRight(
   const int  right      // размер отступа
   )
```

Параметры

right

[in] Размер отступа в пикселях.

© 2000-2025, MetaQuotes Ltd.

IndentUp (Метод Get)

Возвращает отступ графика от верхней границы.

```
int  IndentUp()
```

Возвращаемое значение

Размер отступа в пикселях.

IndentUp (Метод Set)

Устанавливает отступ графика от верхней границы.

```
void  IndentUp(
   const int  up      // размер отступа
   )
```

Параметры

up

[in] Значение отступа в пикселях.

© 2000-2025, MetaQuotes Ltd.

Redraw

Перерисовывает график.

```
bool  Redraw(
   const bool  rescale=false      // значение флага
   )
```

Параметры

rescale=false

[in] Флаг, указывающий, нужно ли выполнить перемасштабирование графика.

Возвращаемое значение

true в случае удачи, иначе false.

© 2000-2025, MetaQuotes Ltd.

ResetParameters

Сбрасывает параметры, необходимые для перерисовки графика.

```
void  ResetParameters()
```

© 2000-2025, MetaQuotes Ltd.

ScaleX

Масштабирует значение по оси X.

```
virtual int  ScaleX(
   double  x      // значение по оси X
   )
```

Параметры

x

[in] Реальное значение по оси X.

Возвращаемое значение

Значение в пикселях.

Примечание

Масштабирует реальное значение в пиксели для отображения на графике.

© 2000-2025, MetaQuotes Ltd.

ScaleY

Масштабирует значение по оси Y.

```
virtual int  ScaleY(
   double  y      // значение по оси y
   )
```

Параметры

y

[in] Реальное значение по оси y.

Возвращаемое значение

Значение в пикселях.

Примечание

Масштабирует реальное значение в пиксели для отображения на графике.

© 2000-2025, MetaQuotes Ltd.

SetDefaultParameters

Устанавливает параметры графика в значения по умолчанию.

```
void  SetDefaultParameters()
```

© 2000-2025, MetaQuotes Ltd.

Width

Возвращает ширину графика в пикселях.

```
int  Width()
```

Возвращаемое значение

Ширина графика в пикселях.

© 2000-2025, MetaQuotes Ltd.

Технические индикаторы и таймсерии

Этот раздел содержит детали работы с классами технических индикаторов и таймсерий и, описание соответствующих компонентов стандартной библиотеки MQL5.

Использование классов технических индикаторов и таймсерий позволит сэкономить время при создании пользовательских программ (скриптов, экспертов).

Стандартная библиотека MQL5 (в части технических индикаторов и таймсерий) размещается в рабочем каталоге терминала в папке Include\Indicators.

Класс/группа	Описание
Базовые классы	Группа базовых и вспомогательных классов
Таймсерии	Группа классов "Таймсерии"
Индикаторы тренда	Группа классов "Тренд"
Осцилляторы	Группа классов "Осциляторы"
Индикаторы объема	Группа классов "Объемы"
Индикаторы Билла Вильямса	Группа классов "Билл Вильямс"
Пользовательский индикатор	Класс "Пользовательский индикатор"

© 2000-2025, MetaQuotes Ltd.

Группа базовых и вспомогательных классов технических индикаторов и таймсерий

Этот раздел содержит детали работы с группой базовых и вспомогательных классов технических индикаторов и таймсерий и описание соответствующих компонентов стандартной библиотеки MQL5.

Класс/группа	Описание
CSpreadBuffer	Класс буфера истории спредов
CTimeBuffer	Класс буфера времени открытия баров
CTickVolumeBuffer	Класс буфера тиковых объемов баров
CRealVolumeBuffer	Класс буфера реальных объемов баров
CDoubleBuffer	Базовый класс буфера данных типа double
COpenBuffer	Класс буфера цен открытия баров
CHighBuffer	Класс буфера максимальных цен баров
CLowBuffer	Класс буфера минимальных цен баров
CCloseBuffer	Класс буфера цен закрытия баров
CIndicatorBuffer	Класс буфера технического индикатора
CSeries	Базовый класс доступа к серийным данным
CPriceSeries	Базовый класс доступа к ценовым данным
CIndicator	Базовый класс технического индикатора
CIndicators	Коллекция технических индикаторов и таймсерий

Методы унаследованные от 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

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Класс CSpreadBuffer

Класс CSpreadBuffer является классом для упрощенного доступа к историческим данным спредов.

Описание

Класс CSpreadBuffer обеспечивает упрощенный доступ к историческим данным спредов.

Декларация

```
class CSpreadBuffer: public CArrayInt
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
CArray
CArrayInt
CSpreadBuffer

Методы класса по группам

Атрибуты	
Size	Устанавливает размер буфера
Настройки	
SetSymbolPeriod	Устанавливает рабочие символ и период
Доступ к данным	
At	Получает элемент буфера
Обновление данных	
virtual Refresh	Освежает весь буфер
virtual RefreshCurrent	Освежает только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayInt

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

Устанавливает размер буфера.

```
void  Size(
   const int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Устанавливает рабочие символ и период.

```
void  SetSymbolPeriod(
   const string           symbol,      // символ
   const ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Новый рабочий символ.

period

[in] Новый рабочий период (значение перечисления ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент буфера.

```
int  At(
   const int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
virtual bool  Refresh()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
virtual bool  RefreshCurrent()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс CTimeBuffer

Класс CTimeBuffer является классом для упрощенного доступа к историческим данным времени открытия баров.

Описание

Класс CTimeBuffer обеспечивает упрощенный доступ к историческим данным времени открытия баров.

Декларация

```
class CTimeBuffer: public CArrayLong
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayLong
      CTimeBuffer

Методы класса по группам

Атрибуты	
Size	Устанавливает размер буфера
Настройки	
SetSymbolPeriod	Устанавливает рабочие символ и период
Доступ к данным	
At	Получает элемент буфера
Обновление данных	
virtual Refresh	Обновляет весь буфер
virtual RefreshCurrent	Обновляет только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayLong

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

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

© 2000-2025, MetaQuotes Ltd.

Size

Устанавливает размер буфера.

```
void  Size(
   const int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Устанавливает рабочие символ и период.

```
void  SetSymbolPeriod(
   const string           symbol,      // символ
   const ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Новый рабочий символ.

period

[in] Новый рабочий период (значение перечисления ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент буфера.

```
long  At(
   const int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
virtual bool  Refresh()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
virtual bool  RefreshCurrent()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс CTickVolumeBuffer

Класс CTickVolumeBuffer является классом для упрощенного доступа к историческим данным тиковых объёмов баров.

Описание

Класс CTickVolumeBuffer обеспечивает упрощенный доступ к историческим данным тиковых объёмов баров.

Декларация

```
class CTickVolumeBuffer: public CArrayLong
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayLong
      CTickVolumeBuffer

Методы класса по группам

Атрибуты	
Size	Устанавливает размер буфера
Настройки	
SetSymbolPeriod	Устанавливает рабочие символ и период
Доступ к данным	
At	Получает элемент буфера
Обновление данных	
virtual Refresh	Обновляет весь буфер
virtual RefreshCurrent	Обновляет только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayLong

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

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

© 2000-2025, MetaQuotes Ltd.

Size

Устанавливает размер буфера.

```
void  Size(
   const int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Устанавливает рабочие символ и период.

```
void  SetSymbolPeriod(
   const string           symbol,      // символ
   const ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Новый рабочий символ.

period

[in] Новый рабочий период (значение перечисления ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент буфера.

```
long  At(
   const int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
virtual bool  Refresh()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
virtual bool  RefreshCurrent()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс CRealVolumeBuffer

Класс CRealVolumeBuffer является классом для упрощенного доступа к историческим данным реальных объёмов баров.

Описание

Класс CRealVolumeBuffer обеспечивает упрощенный доступ к историческим данным реальных объёмов баров.

Декларация

```
class CRealVolumeBuffer: public CArrayLong
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayLong
      CRealVolumeBuffer

Методы класса по группам

Атрибуты	
Size	Устанавливает размер буфера
Настройки	
SetSymbolPeriod	Устанавливает рабочие символ и период
Доступ к данным	
At	Получает элемент буфера
Обновление данных	
virtual Refresh	Обновляет весь буфер
virtual RefreshCurrent	Обновляет только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayLong

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

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

© 2000-2025, MetaQuotes Ltd.

Size

Устанавливает размер буфера.

```
void  Size(
   const int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Устанавливает рабочие символ и период.

```
void  SetSymbolPeriod(
   const string           symbol,      // символ
   const ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Новый рабочий символ.

period

[in] Новый рабочий период (значение перечисления ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент буфера.

```
long  At(
   const int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
virtual bool  Refresh()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
virtual bool  RefreshCurrent()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс CDoubleBuffer

Класс CDoubleBuffer является базовым классом для упрощенного доступа к данным буфера типа double.

Описание

Класс CDoubleBuffer обеспечивает своим потомкам возможность упрощенного доступа к данным буфера типа double.

Декларация

```
class CDoubleBuffer: public CArrayDouble
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayDouble
      CDoubleBuffer

Прямые потомки

CCloseBuffer, CHighBuffer, CIndicatorBuffer, CLowBuffer, COpenBuffer

Методы класса по группам

Атрибуты	
Size	Устанавливает размер буфера
Настройки	
SetSymbolPeriod	Устанавливает рабочие символ и период
Доступ к данным	
At	Получает элемент буфера
Обновление данных	
virtual Refresh	Обновляет весь буфер
virtual RefreshCurrent	Обновляет только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Size

Устанавливает размер буфера.

```
void  Size(
   const int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

Устанавливает рабочие символ и период.

```
void  SetSymbolPeriod(
   const string           symbol,      // символ
   const ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Новый рабочий символ.

period

[in] Новый рабочий период (значение перечисления ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент буфера.

```
double  At(
   const int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
virtual bool  Refresh()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
virtual bool  RefreshCurrent()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс COpenBuffer

Класс COpenBuffer является классом для упрощенного доступа к историческим данным цен открытия баров.

Описание

Класс COpenBuffer обеспечивает упрощенный доступ к историческим данным цен открытия баров.

Декларация

```
class COpenBuffer: public CDoubleBuffer
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        COpenBuffer

Методы класса по группам

Обновление данных	
virtual Refresh	Обновляет весь буфер
virtual RefreshCurrent	Обновляет только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
virtual bool  Refresh()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
virtual bool  RefreshCurrent()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс CHighBuffer

Класс CHighBuffer является классом для упрощенного доступа к историческим данным максимальных цен баров.

Описание

Класс CHighBuffer обеспечивает упрощенный доступ к историческим данным максимальных цен баров.

Декларация

```
class CHighBuffer: public CDoubleBuffer
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CHighBuffer

Методы класса по группам

Обновление данных	
virtual Refresh	Обновляет весь буфер
virtual RefreshCurrent	Обновляет только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
virtual bool  Refresh()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
virtual bool  RefreshCurrent()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс CLowBuffer

Класс CLowBuffer является классом для упрощенного доступа к историческим данным минимальных цен баров.

Описание

Класс CLowBuffer обеспечивает упрощенный доступ к историческим данным минимальных цен баров.

Декларация

```
class CLowBuffer: public CDoubleBuffer
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CLowBuffer

Методы класса по группам

Обновление данных	
virtual Refresh	Обновляет весь буфер
virtual RefreshCurrent	Обновляет только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
virtual bool  Refresh()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
virtual bool  RefreshCurrent()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс CCloseBuffer

Класс CCloseBuffer является классом для упрощенного доступа к историческим данным цен закрытия баров.

Описание

Класс CCloseBuffer обеспечивает упрощенный доступ к историческим данным цен закрытия баров.

Декларация

```
class CCloseBuffer: public CDoubleBuffer
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CCloseBuffer

Методы класса по группам

Обновление данных	
virtual Refresh	Обновляет весь буфер
virtual RefreshCurrent	Обновляет только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
virtual bool  Refresh()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
virtual bool  RefreshCurrent()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс CIndicatorBuffer

Класс CIndicatorBuffer является классом для упрощенного доступа к данным буфера технического индикатора.

Описание

Класс CIndicatorBuffer обеспечивает упрощенный доступ к данным буфера технического индикатора.

Декларация

```
class CIndicatorBuffer: public CDoubleBuffer
```

Заголовок

```
#include <Indicators\Indicator.mqh>
```

Иерархия наследования

CObject
CArray
CArrayDouble
CDoubleBuffer
CIndicatorBuffer

Методы класса по группам

Атрибуты	
Offset	Получает/устанавливает смещение буфера
Name	Получает/устанавливает наименование буфера
Доступ к данным	
At	Получает элемент буфера
Обновление данных	
Refresh	Обновляет весь буфер
RefreshCurrent	Обновляет только текущее значение

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayDouble

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

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

Методы унаследованные от CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Offset

Получает смещение буфера.

```
int  Offset() const
```

Возвращаемое значение

Смещение буфера.

Offset

Устанавливает смещение буфера.

```
void  Offset(
   const int  offset      // смещение
   )
```

Параметры

offset

[in] Новое смещение буфера.

© 2000-2025, MetaQuotes Ltd.

Name

Получает наименование буфера.

```
string  Name() const
```

Возвращаемое значение

Наименование буфера.

Name

Устанавливает наименование буфера.

```
void  Name(
   const string  name      // наименование
   )
```

Параметры

name

[in] Новое наименование буфера.

© 2000-2025, MetaQuotes Ltd.

At

Получает элемент буфера.

```
double  At(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента в буфере.

Возвращаемое значение

Элемент буфера по указанному индексу.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет весь буфер.

```
bool  Refresh(
   const int  handle,     // хэндл индикатора
   const int  num         // номер буфера
   )
```

Параметры

handle

[in] Хэндл индикатора.

num

[in] Номер индикаторного буфера.

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Обновляет нулевой элемент буфера.

```
bool  RefreshCurrent(
   const int  handle,     // хэндл индикатора
   const int  num         // номер буфера
   )
```

Параметры

handle

[in] Хэндл индикатора.

num

[in] Номер индикаторного буфера.

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить буфер.

© 2000-2025, MetaQuotes Ltd.

Класс CSeries

Класс CSeries является базовым классом для классов доступа к серийным данным стандартной библиотеки MQL5.

Описание

Класс CSeries обеспечивает всем своим потомкам (классам таймсерий и индикаторов) возможность упрощенного доступа к общим функциям API MQL5 в части работы с серийными данными.

Декларация

```
class CSeries: public CArrayObj
```

Заголовок

```
#include <Indicators\Series.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries

Прямые потомки

CIndicator, CiRealVolume, CiSpread, CiTickVolume, CiTime, CPriceSeries

Методы класса по группам

Атрибуты	
Name	Получает имя таймсерии или индикатора
BuffersTotal	Получает количество буферов таймсерии или индикатора
Timeframe	Получает флаг таймфрейма таймсерии или индикатора
Symbol	Получает имя рабочего инструмента таймсерии или индикатора
Period	Получает рабочий таймфрейм таймсерии или индикатора
RefreshCurrent	Устанавливает/сбрасывает флаг обновления текущих данных
Доступ к данным	
virtual BufferResize	Устанавливает размеры буферов таймсерии или индикатора
Обновление данных	
virtual Refresh	Обновляет данные таймсерии или индикатора

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
PeriodDescription	Преобразует ENUM_TIMEFRAMES в строку

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Name

Получает имя таймсерии или индикатора.

```
string  Name() const
```

Возвращаемое значение

Имя таймсерии или индикатора.

© 2000-2025, MetaQuotes Ltd.

BuffersTotal

Получает количество буферов таймсерии или индикатора.

```
int  BuffersTotal() const
```

Возвращаемое значение

Количество буферов таймсерии или индикатора.

Примечание

Количество буферов таймсерии всегда 1.

© 2000-2025, MetaQuotes Ltd.

Timeframe

Получает флаг таймфрейма таймсерии или индикатора.

```
int  Timeframe() const
```

Возвращаемое значение

Флаг таймфрейма таймсерии или индикатора.

Примечание

Формируется как флаги видимости объектов.

© 2000-2025, MetaQuotes Ltd.

Symbol

Получает имя рабочего инструмента таймсерии или индикатора.

```
string  Symbol() const
```

Возвращаемое значение

Имя рабочего инструмента (символа) таймсерии или индикатора.

© 2000-2025, MetaQuotes Ltd.

Period

Получает рабочий таймфрейм таймсерии или индикатора.

```
ENUM_TIMEFRAMES  Period() const
```

Возвращаемое значение

Рабочий таймфрейм таймсерии или индикатора (значение перечисления ENUM_TIMEFRAMES).

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

Устанавливает флаг необходимости постоянного обновления текущих значений таймсерии или индикатора.

```
string  RefreshCurrent(
   const bool  flag      // значение
   )
```

Параметры

flag

[in] Новое значение флага.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

BufferSize

Получает количество доступных данных в буфере таймсерии или индикатора.

```
int  BufferSize() const
```

Возвращаемое значение

Количество доступных данных в буфере таймсерии или индикатора.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Устанавливает размеры буферов таймсерии или индикатора.

```
virtual bool  BufferResize(
   const int  size      // размер
   )
```

Параметры

size

[in] Новый размер буферов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Все буферы индикатора или таймсерии будут иметь одинаковый размер.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет данные таймсерии или индикатора.

```
virtual void  Refresh(
   const int  flags      // флаги
   )
```

Параметры

flags

[in] Флаги обновления таймфреймов.

© 2000-2025, MetaQuotes Ltd.

PeriodDescription

Преобразует в строку значение перечисления ENUM_TIMEFRAMES.

```
string  PeriodDescription(
   const int  val=0      // значение
   )
```

Параметры

val=0

[in] Значение для преобразования.

Возвращаемое значение

Строка, соответствующая значению перечисления ENUM_TIMEFRAMES.

Примечание

Если значение не указано или равно нулю, то в строку преобразуется рабочий таймфрейм таймсерии или индикатора.

© 2000-2025, MetaQuotes Ltd.

Класс CPriceSeries

Класс CPriceSeries является базовым классом для классов доступа к ценовым данным.

Описание

Класс CPriceSeries обеспечивает всем своим потомкам возможность упрощенного доступа к общим функциям API MQL5 в части работы с ценовыми данными.

Декларация

```
class CPriceSeries: public CSeries
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries

Прямые потомки

CiClose, CiHigh, CiLow, CiOpen

Методы класса по группам

Создание	
virtual BufferResize	Устанавливает размеры буфера серии
Доступ к данным	
virtual GetData	Получает указанный элемент буфера серии
Обновление данных	
virtual Refresh	Обновляет данные серии
Поиск экстремумов	
virtual MinIndex	Получает индекс минимального значения в указанном диапазоне
virtual MinValue	Получает минимальное значение в указанном диапазоне
virtual MaxIndex	Получает индекс максимального значения в указанном диапазоне
virtual MaxValue	Получает максимальное значение в указанном диапазоне

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

BufferResize

Устанавливает размеры буфера серии.

```
virtual bool  BufferResize(
   const int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

© 2000-2025, MetaQuotes Ltd.

GetData

Получает указанный элемент буфера серии.

```
double  GetData(
   const int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера серии, либо EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет данные серии.

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // флаги
   )
```

Параметры

flags=OBJ_ALL_PERIODS

[in] Флаги таймфреймов.

© 2000-2025, MetaQuotes Ltd.

MinIndex

Получает индекс минимального значения в указанном интервале.

```
virtual int  MinIndex(
   const int  start,     // размер
   constint  count       // количество
   ) const
```

Параметры

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

Возвращаемое значение

Индекс минимального значения буфера серии в указанном интервале, либо -1.

© 2000-2025, MetaQuotes Ltd.

MinValue

Получает минимальное значение в указанном интервале.

```
virtual double  MinValue(
   const int    start,      // размер
   const int    count,      // количество
   int&         index       // ссылка
   ) const
```

Параметры

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

index

[out] Ссылка на переменную для размещения значения индекса найденного элемента.

Возвращаемое значение

Минимальное значение буфера серии в указанном интервале, либо EMPTY_VALUE.

Примечание

Индекс найденного элемента помещается по ссылке index.

© 2000-2025, MetaQuotes Ltd.

MaxIndex

Получает индекс максимального значения в указанном интервале.

```
virtual int  MaxIndex(
   const int  start,      // индекс
   const int  count       // количество
   ) const
```

Параметры

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

Возвращаемое значение

Индекс максимального значения буфера серии в указанном интервале, либо -1.

© 2000-2025, MetaQuotes Ltd.

MaxValue

Получает максимальное значение в указанном интервале.

```
virtual double  MaxValue(
   const int    start,      // размер
   const int    count,      // количество
   int&         index       // ссылка
   ) const
```

Параметры

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

index

[out] Ссылка на переменную для размещения значения индекса найденного элемента.

Возвращаемое значение

Максимальное значение буфера серии в указанном интервале, либо EMPTY_VALUE.

Примечание

Индекс найденного элемента помещается по ссылке index.

© 2000-2025, MetaQuotes Ltd.

Класс CIndicator

Класс CIndicator является базовым классом для классов технических индикаторов стандартной библиотеки MQL5.

Описание

Класс CIndicator обеспечивает всем своим потомкам возможность упрощенного доступа к общим функциям API MQL5 в части работы с техническими индикаторами.

Декларация

```
class CIndicator: public CSeries
```

Заголовок

```
#include <Indicators\Indicator.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator

Прямые потомки

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

Методы класса по группам

Атрибуты	
Handle	Получает хэндл индикатора
Status	Получает статус создания индикатора
FullRelease	Устанавливает флаг освобождения хендла
Создание	
Create	Создает индикатор
BufferResize	Устанавливает размеры буферов индикатора
Доступ к данным	
GetData	Копирует данные из буфера индикатора
Обновление данных	

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
Refresh	Обновляет данные индикатора
Поиск экстремумов	
Minimum	Получает индекс минимального значения в указанном диапазоне
MinValue	Получает минимальное значение в указанном диапазоне
Maximum	Получает индекс максимального значения в указанном диапазоне
MaxValue	Получает максимальное значение в указанном диапазоне
Преобразование перечислений	
MethodDescription	Преобразует ENUM_MA_METHOD в строку
PriceDescription	Преобразует ENUM_APPLIED_PRICE в строку
VolumeDescription	Преобразует ENUM_APPLIED_VOLUME в строку
Работа с графиком	
AddToChart	Добавляет индикатор на график
DeleteFromChart	Удаляет индикатор с графика

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Handle

Получает хэндл индикатора.

```
int  Handle() const
```

Возвращаемое значение

Хэндл индикатора.

© 2000-2025, MetaQuotes Ltd.

Status

Получает статус создания индикатора.

```
string  Status() const
```

Возвращаемое значение

Статус создания индикатора.

© 2000-2025, MetaQuotes Ltd.

FullRelease

Устанавливает флаг освобождения хендла.

```
void  FullRelease(
   const bool  flag=true      // флаг
   )
```

Параметры

flag

[in] Новое значение флага освобождения хендла.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор указанного типа с указанными параметрами.

```
bool  Create(
   const string           symbol,          // имя символа
   const ENUM_TIMEFRAMES  period,          // период
   const ENUM_INDICATOR   type,            // тип
   const int              num_params,      // количество параметров
   const MqlParam&        params[]         // массив параметров
   )
```

Параметры

symbol

[in] Имя символа индикатора.

period

[in] Таймфрейм индикатора (значение перечисления ENUM_TIMEFRAMES).

type

[in] Тип индикатора (значение перечисления ENUM_INDICATOR).

num_params

[in] Количество параметров для создания индикатора.

params

[in] Ссылка на массив параметров для создания индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Устанавливает размеры буферов индикатора.

```
virtual bool  BufferResize(
   const int  size      // размер
   )
```

Параметры

size

[in] Новый размер буферов.

Возвращаемое значение

Возвращает true в случае успеха, иначе false.

Примечание

Все буферы индикатора будут иметь одинаковый размер.

© 2000-2025, MetaQuotes Ltd.

BarsCalculated

Возвращает количество рассчитанных данных для индикатора.

```
int  BarsCalculated() const;
```

Возвращаемое значение

Возвращает количество рассчитанных данных в индикаторном буфере или -1 в случае ошибки (данные еще не рассчитаны).

© 2000-2025, MetaQuotes Ltd.

GetData

Получает указанный элемент указанного буфера индикатора. Для работы с актуальными данными перед использованием необходимо вызвать Refresh().

```
double  GetData(
   const int  buffer_num,     // номер буфера
   const int  index           // индекс
   ) const
```

Параметры

buffer_num

[in] Номер буфера индикатора.

index

[in] Индекс элемента буфера индикатора.

Возвращаемое значение

значение - в случае удачи, EMPTY_VALUE - если не удалось получить данные.

GetData

Получает данные из буфера индикатора по стартовой позиции и количеству.

```
int  GetData(
   const int      start_pos,      // позиция
   const int      count,          // количество
   const int      buffer_num,     // номер буфера
   double&        buffer[]        // массив
   ) const
```

Параметры

start_pos

[in] Стартовая позиция буфера индикатора.

count

[in] Количество элементов буфера индикатора.

buffer_num

[in] Номер буфера индикатора.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

Количество полученных значений индикатора из указанного буфера в случае удачи, иначе -1.

GetData

Получает данные из буфера индикатора по начальному времени и количеству.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   const datetime  start_time,      // начальное время
   const int       count,           // количество
   const int       buffer_num,      // номер буфера
   double&         buffer[]         // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера индикатора.

count

[in] Количество элементов буфера индикатора.

buffer_num

[in] Номер буфера индикатора.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

Количество полученных значений индикатора из указанного буфера в случае удачи, иначе -1.

GetData

Получает данные из буфера индикатора по начальному и конечному времени.

```
int  GetData(
   const datetime  start_time,      // начальное время
   const datetime  stop_time,       // конечное время
   const int       buffer_num,      // номер буфера
   double&         buffer[]         // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера индикатора.

stop_time

[in] Время конечного элемента буфера индикатора.

buffer_num

[in] Номер буфера индикатора.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

Количество полученных значений индикатора из указанного буфера в случае удачи, иначе -1.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет данные индикатора. Рекомендуется вызывать перед использованием GetData().

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // флаги
   )
```

Параметры

flags=OBJ_ALL_PERIODS

[in] Флаги обновления таймфреймов.

© 2000-2025, MetaQuotes Ltd.

Minimum

Получает индекс минимального элемента указанного буфера, в указанном интервале.

```
int  Minimum(
   const int  buffer_num,      // номер буфера
   const int  start,           // стартовый индекс
   const int  count            // количество
   ) const
```

Параметры

buffer_num

[in] Номер буфера, для поиска значения.

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

Возвращаемое значение

Индекс минимального элемента указанного буфера, в указанном интервале.

© 2000-2025, MetaQuotes Ltd.

MinValue

Получает значение минимального элемента указанного буфера, в указанном интервале.

```
double  MinValue(
   const int   buffer_num,     // номер буфера
   const int   start,          // стартовый индекс
   const int   count,          // количество
   int&        index           // ссылка
   ) const
```

Параметры

buffer_num

[in] Номер буфера, для поиска значения.

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

index

[out] Ссылка на переменную, для размещения значения индекса найденного элемента.

Возвращаемое значение

Значение минимального элемента указанного буфера, в указанном интервале.

Примечание

Индекс найденного элемента помещается по ссылке index.

© 2000-2025, MetaQuotes Ltd.

Maximum

Получает индекс максимального элемента указанного буфера, в указанном интервале.

```
int  Maximum(
   const int  buffer_num,      // номер буфера
   const int  start,           // стартовый индекс
   const int  count            // количество
   ) const
```

Параметры

buffer_num

[in] Номер буфера, для поиска значения.

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

Возвращаемое значение

Индекс максимального элемента указанного буфера, в указанном интервале.

© 2000-2025, MetaQuotes Ltd.

MaxValue

Получает значение максимального элемента указанного буфера, в указанном интервале.

```
double  MaxValue(
   const int   buffer_num,     // номер буфера
   const int   start,          // стартовый индекс
   const int   count,          // количество
   int&        index           // ссылка
   ) const
```

Параметры

buffer_num

[in] Номер буфера, для поиска значения.

start

[in] Стартовый индекс интервала поиска.

count

[in] Размер интервала поиска (количество элементов).

index

[out] Ссылка на переменную, для размещения значения индекса найденного элемента.

Возвращаемое значение

Значение максимального элемента указанного буфера, в указанном интервале.

Примечание

Индекс найденного элемента помещается по ссылке index.

© 2000-2025, MetaQuotes Ltd.

MethodDescription

Преобразует в строку значение перечисления ENUM_MA_METHOD.

```
string  MethodDescription(
   const int  val      // значение
   ) const
```

Параметры

val

[in] Значение для преобразования.

Возвращаемое значение

Строка, соответствующая значению перечисления ENUM_MA_METHOD.

© 2000-2025, MetaQuotes Ltd.

PriceDescription

Преобразует в строку значение перечисления ENUM_APPLIED_PRICE.

```
string  PriceDescription(
   const int  val      // значение
   ) const
```

Параметры

val

[in] Значение для преобразования.

Возвращаемое значение

Строка, соответствующая значению перечисления ENUM_APPLIED_PRICE.

© 2000-2025, MetaQuotes Ltd.

VolumeDescription

Преобразует в строку значение перечисления ENUM_APPLIED_VOLUME.

```
string  VolumeDescription(
   const int  val      // значение
   ) const
```

Параметры

val

[in] Значение для преобразования.

Возвращаемое значение

Строка, соответствующая значению перечисления ENUM_APPLIED_VOLUME.

© 2000-2025, MetaQuotes Ltd.

AddToChart

Добавляет индикатор на график.

```
bool  AddToChart(
   const long  chart,      // идентификатор графика
   const int   subwin      // индекс подокна
   )
```

Параметры

chart

[in] Идентификатор графика.

subwin

[in] Индекс подокна графика.

Возвращаемое значение

true - в случае удачи, false - если не удалось добавить индикатор на график.

© 2000-2025, MetaQuotes Ltd.

DeleteFromChart

Удаляет индикатор с графика.

```
bool  DeleteFromChart(
   const long  chart,      // идентификатор графика
   const int  subwin       // индекс подокна
   )
```

Параметры

chart

[in] Идентификатор графика.

subwin

[in] Индекс подокна графика.

Возвращаемое значение

true - в случае удачи, false - если не удалось удалить индикатор с графика.

© 2000-2025, MetaQuotes Ltd.

Класс CIndicators

Класс CIndicators является классом для коллекционирования экземпляров классов таймсерий и технических индикаторов.

Описание

Класс CIndicators обеспечивает создание экземпляров классов технических индикаторов, их хранение и управление ими (синхронизацию данных, управление хэндлами и памятью).

Декларация

```
class CIndicators: public CArrayObj
```

Заголовок

```
#include <Indicators\Indicators.mqh>
```

Методы класса по группам

Создание	
Create	Создает индикатор
Обновление данных	
Refresh	Обновляет данные индикатора

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор указанного типа с указанными параметрами.

```
CIndicator*  Create(
   const string           symbol,     // имя символа
   const ENUM_TIMEFRAMES  period,     // период
   const ENUM_INDICATOR   type,       // тип
   const int              count,      // количество параметров
   const MqlParam&        params[]    // массив параметров
   )
```

Параметры

symbol

[in] Имя символа индикатора.

period

[in] Таймфрейм индикатора (значение перечисления ENUM_TIMEFRAMES).

type

[in] Тип индикатора (значение перечисления ENUM_INDICATOR).

count

[in] Количество параметров для создания индикатора.

params

[in] Ссылка на массив параметров для создания индикатора.

Возвращаемое значение

Указатель на созданный индикатор - в случае удачи, NULL - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет данные всех таймсерий и технических индикаторов коллекции.

```
int  Refresh()
```

Возвращаемое значение

Флаги обновленных таймфреймов (формируется как флаги видимости объектов.)

© 2000-2025, MetaQuotes Ltd.

Классы для работы таймсериями

Этот раздел содержит описание классов для работы таймсериями и описание соответствующих компонентов стандартной библиотеки MQL5.

Класс	Описание
CiSpread	Класс доступа к истории спредов
CiTime	Класс доступа к времени открытия баров
CiTickVolume	Класс доступа к тиковым объемам баров
CiRealVolume	Класс доступа к реальным объемам баров
CiOpen	Класс доступа к ценам открытия баров
CiHigh	Класс доступа к максимальным ценам баров
CiLow	Класс доступа к минимальным ценам баров
CiClose	Класс доступа к ценам закрытия баров

© 2000-2025, MetaQuotes Ltd.

Класс CiSpread

Класс CiSpread является классом для доступа к сериям истории спредов.

Описание

Класс CiSpread обеспечивает доступ к сериям истории спредов.

Декларация

```
class CiSpread: public CSeries
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CiSpread

Методы класса по группам

Создание	
Create	Создает серию
BufferResize	Устанавливает размеры буфера серии
Доступ к данным	
GetData	Получает данные серии
Обновление данных	
Refresh	Обновляет данные серии

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

Создает таймсерию для доступа к истории спредов с указанными параметрами.

```
bool  Create(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Имя символа таймсерии.

period

[in] Таймфрейм таймсерии (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать таймсерию.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Устанавливает размеры буфера серии.

```
virtual void  BufferResize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

© 2000-2025, MetaQuotes Ltd.

GetData

Получает указанный элемент буфера серии.

```
int  GetData(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера серии, либо 0.

GetData

Получает данные из буфера таймсерии по стартовой позиции и количеству.

```
int  GetData(
   int   start_pos,      // позиция
   int   count,          // количество
   int&  buffer          // массив
   ) const
```

Параметры

start_pos

[in] Стартовая позиция буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному времени и количеству.

```
int  GetData(
   datetime  start_time,      // начальное время
   int       count,           // количество
   int&      buffer           // массив
   ) const
```

Параметры

start_time

© 2000-2025, MetaQuotes Ltd.

[in] Время начального элемента буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному и конечному времени.

```
int  GetData(
   datetime  start_time,      // начальное время
   datetime  stop_time,       // конечное время
   int&      buffer           // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

stop_time

[in] Время конечного элемента буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет данные серии.

```
virtual void  Refresh(
   int  flags      // флаги
   )
```

Параметры

flags

[in] Флаги таймфреймов.

© 2000-2025, MetaQuotes Ltd.

Класс CiTime

Класс CiTime является классом для доступа к сериям времени открытия баров.

Описание

Класс CiTime обеспечивает доступ к сериям времени открытия баров.

Декларация

```
class CiTime: public CSeries
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CiTime

Методы класса по группам

Создание	
Create	Создает серию
BufferResize	Устанавливает размеры буфера серии
Доступ к данным	
GetData	Получает данные серии
Обновление данных	
Refresh	Обновляет данные серии

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

Создает таймсерию для доступа к времени открытия баров с указанными параметрами.

```
bool  Create(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Имя символа таймсерии.

period

[in] Таймфрейм таймсерии (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать таймсерию.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Устанавливает размеры буфера серии.

```
virtual void  BufferResize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

© 2000-2025, MetaQuotes Ltd.

GetData

Получает указанный элемент буфера серии.

```
datetime  GetData(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера серии, либо 0.

GetData

Получает данные из буфера таймсерии по стартовой позиции и количеству.

```
int  GetData(
   int    start_pos,      // позиция
   int    count,          // количество
   long&  buffer          // массив
   ) const
```

Параметры

start_pos

[in] Стартовая позиция буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному времени и количеству.

```
int  GetData(
   datetime  start_time,      // начальное время
   int       count,           // количество
   long&     buffer           // массив
   ) const
```

Параметры

start_time

© 2000-2025, MetaQuotes Ltd.

[in] Время начального элемента буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному и конечному времени.

```
int  GetData(
   datetime  start_time,      // начальное время
   datetime  stop_time,       // конечное время
   long&     buffer           // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

stop_time

[in] Время конечного элемента буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет данные серии.

```
virtual void  Refresh(
   int  flags      // флаги
   )
```

Параметры

flags

[in] Флаги таймфреймов.

© 2000-2025, MetaQuotes Ltd.

Класс CiTickVolume

Класс CiTickVolume является классом для доступа к сериям тиковых объёмов баров.

Описание

Класс CiTickVolume обеспечивает доступ к сериям тиковых объёмов баров.

Декларация

```
class CiTickVolume: public CSeries
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CiTickVolume

Методы класса по группам

Создание	
Create	Создает серию
BufferResize	Устанавливает размеры буфера серии
Доступ к данным	
GetData	Получает данные серии
Обновление данных	
Refresh	Обновляет данные серии

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

Создает таймсерию для доступа к тиковым объёмам баров с указанными параметрами.

```
bool  Create(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Имя символа таймсерии.

period

[in] Таймфрейм таймсерии (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать таймсерию.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Устанавливает размеры буфера серии.

```
virtual void  BufferResize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

© 2000-2025, MetaQuotes Ltd.

GetData

Получает указанный элемент буфера серии.

```
long  GetData(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера серии, либо 0.

GetData

Получает данные из буфера таймсерии по стартовой позиции и количеству.

```
int  GetData(
   int    start_pos,      // позиция
   int    count,          // количество
   long&  buffer          // массив
   ) const
```

Параметры

start_pos

[in] Стартовая позиция буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному времени и количеству.

```
int  GetData(
   datetime  start_time,      // начальное время
   int       count,           // количество
   long&     buffer           // массив
   ) const
```

Параметры

start_time

© 2000-2025, MetaQuotes Ltd.

[in] Время начального элемента буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному и конечному времени.

```
int  GetData(
   datetime  start_time,      // начальное время
   datetime  stop_time,       // конечное время
   long&     buffer           // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

stop_time

[in] Время конечного элемента буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет данные серии.

```
virtual void  Refresh(
   int  flags      // флаги
   )
```

Параметры

flags

[in] Флаги таймфреймов.

© 2000-2025, MetaQuotes Ltd.

Класс CiRealVolume

Класс CiRealVolume является классом для доступа к сериям реальных объёмов баров.

Описание

Класс CiRealVolume обеспечивает доступ к сериям реальных объёмов баров.

Декларация

```
class CiRealVolume: public CSeries
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CiRealVolume

Методы класса по группам

Создание	
Create	Создает серию
BufferResize	Устанавливает размеры буфера серии
Доступ к данным	
GetData	Получает данные серии
Обновление данных	
Refresh	Обновляет данные серии

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

Создает таймсерию для доступа к реальным объёмам баров с указанными параметрами.

```
bool  Create(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Имя символа таймсерии.

period

[in] Таймфрейм таймсерии (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать таймсерию.

© 2000-2025, MetaQuotes Ltd.

BufferResize

Устанавливает размеры буфера серии.

```
virtual void  BufferResize(
   int  size      // размер
   )
```

Параметры

size

[in] Новый размер буфера.

© 2000-2025, MetaQuotes Ltd.

GetData

Получает указанный элемент буфера серии.

```
long  GetData(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера серии, либо 0.

GetData

Получает данные из буфера таймсерии по стартовой позиции и количеству.

```
int  GetData(
   int    start_pos,      // позиция
   int    count,          // количество
   long&  buffer          // массив
   ) const
```

Параметры

start_pos

[in] Стартовая позиция буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному времени и количеству.

```
int  GetData(
   datetime  start_time,      // начальное время
   int       count,           // количество
   long&     buffer           // массив
   ) const
```

Параметры

start_time

© 2000-2025, MetaQuotes Ltd.

[in] Время начального элемента буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному и конечному времени.

```
int  GetData(
   datetime  start_time,      // начальное время
   datetime  stop_time,       // конечное время
   long&     buffer           // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

stop_time

[in] Время конечного элемента буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет данные серии.

```
virtual void  Refresh(
   int  flags      // флаги
   )
```

Параметры

flags

[in] Флаги таймфреймов.

© 2000-2025, MetaQuotes Ltd.

Класс CiOpen

Класс CiOpen является классом для доступа к сериям цен открытия баров.

Описание

Класс CiOpen обеспечивает доступ к сериям цен открытия баров.

Декларация

```
class CiOpen: public CPriceSeries
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiOpen

Методы класса по группам

Создание	
Create	Создает серию
Доступ к данным	
GetData	Получает данные серии

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

Создает таймсерию для доступа к ценам открытия баров с указанными параметрами.

```
bool  Create(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Имя символа таймсерии.

period

[in] Таймфрейм таймсерии (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать таймсерию.

© 2000-2025, MetaQuotes Ltd.

GetData

Получает данные из буфера таймсерии по стартовой позиции и количеству.

```
int  GetData(
   int      start_pos,      // позиция
   int      count,          // количество
   double&  buffer          // массив
   ) const
```

Параметры

start_pos

[in] Стартовая позиция буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному времени и количеству.

```
int  GetData(
   datetime  start_time,     // начальное время
   int       count,          // количество
   double&   buffer          // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному и конечному времени.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // начальное время
   datetime  stop_time,       // конечное время
   double&   buffer           // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

stop_time

[in] Время конечного элемента буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

© 2000-2025, MetaQuotes Ltd.

Класс CiHigh

Класс CiHigh является классом для доступа к сериям максимальных цен баров.

Описание

Класс CiHigh обеспечивает доступ к сериям максимальных цен баров.

Декларация

```
class CiHigh: public CPriceSeries
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiHigh

Методы класса по группам

Создание	
Create	Создает серию
Доступ к данным	
GetData	Получает данные серии

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

Создает таймсерию для доступа к максимальным ценам баров, с указанными параметрами.

```
bool  Create(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Имя символа таймсерии.

period

[in] Таймфрейм таймсерии (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать таймсерию.

© 2000-2025, MetaQuotes Ltd.

GetData

Получает данные из буфера таймсерии по стартовой позиции и количеству.

```
int  GetData(
   int      start_pos,      // позиция
   int      count,          // количество
   double&  buffer          // массив
   ) const
```

Параметры

start_pos

[in] Стартовая позиция буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному времени и количеству.

```
int  GetData(
   datetime  start_time,     // начальное время
   int       count,          // количество
   double&   buffer          // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному и конечному времени.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // начальное время
   datetime  stop_time,       // конечное время
   double&   buffer           // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

stop_time

[in] Время конечного элемента буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

© 2000-2025, MetaQuotes Ltd.

Класс CiLow

Класс CiLow является классом для доступа к сериям минимальных цен баров.

Описание

Класс CiLow обеспечивает доступ к сериям минимальных цен баров.

Декларация

```
class CiLow: public CPriceSeries
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiLow

Методы класса по группам

Создание	
Create	Создает серию
Доступ к данным	
GetData	Получает данные серии

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

Создает таймсерию для доступа к минимальным ценам баров с указанными параметрами.

```
bool  Create(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Имя символа таймсерии.

period

[in] Таймфрейм таймсерии (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать таймсерию.

© 2000-2025, MetaQuotes Ltd.

GetData

Получает данные из буфера таймсерии по стартовой позиции и количеству.

```
int  GetData(
   int      start_pos,      // позиция
   int      count,          // количество
   double&  buffer          // массив
   ) const
```

Параметры

start_pos

[in] Стартовая позиция буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному времени и количеству.

```
int  GetData(
   datetime  start_time,     // начальное время
   int       count,          // количество
   double&   buffer          // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному и конечному времени.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // начальное время
   datetime  stop_time,       // конечное время
   double&   buffer           // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

stop_time

[in] Время конечного элемента буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

© 2000-2025, MetaQuotes Ltd.

Класс CiClose

Класс CiClose является классом для доступа к сериям цен закрытия баров.

Описание

Класс CiClose обеспечивает доступ к сериям цен закрытия баров.

Декларация

```
class CiClose: public CPriceSeries
```

Заголовок

```
#include <Indicators\TimeSeries.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiClose

Методы класса по группам

Создание	
Create	Создает серию
Доступ к данным	
GetData	Получает данные серии

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

Создает таймсерию для доступа к ценам закрытия баров, с указанными параметрами.

```
bool  Create(
   string           symbol,      // имя символа
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Имя символа таймсерии.

period

[in] Таймфрейм таймсерии (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать таймсерию.

© 2000-2025, MetaQuotes Ltd.

GetData

Получает данные из буфера таймсерии по стартовой позиции и количеству.

```
int  GetData(
   int      start_pos,      // позиция
   int      count,          // количество
   double&  buffer          // массив
   ) const
```

Параметры

start_pos

[in] Стартовая позиция буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному времени и количеству.

```
int  GetData(
   datetime  start_time,     // начальное время
   int       count,          // количество
   double&   buffer          // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

count

[in] Количество элементов буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

GetData

Получает данные из буфера таймсерии по начальному и конечному времени.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // начальное время
   datetime  stop_time,       // конечное время
   double&   buffer           // массив
   ) const
```

Параметры

start_time

[in] Время начального элемента буфера таймсерии.

stop_time

[in] Время конечного элемента буфера таймсерии.

buffer

[in] Ссылка на массив для размещения данных.

Возвращаемое значение

>=0 - в случае удачи, -1 - если не удалось получить данные.

© 2000-2025, MetaQuotes Ltd.

Группа базовых и вспомогательных классов технических индикаторов и таймсерий

Этот раздел содержит детали работы с группой базовых и вспомогательных классов технических индикаторов и таймсерий и описание соответствующих компонентов стандартной библиотеки MQL5.

Класс/группа	Описание
CiADX	Индикатор "Average Directional Index"
CiADXWilder	Индикатор "Average Directional Index by Welles Wilder"
CiBands	Индикатор "Bollinger Bands®"
CiEnvelopes	Индикатор "Envelopes"
CiIchimoku	Индикатор "Ichimoku Kinko Hyo"
CiMA	Индикатор "Moving Average"
CiSAR	Индикатор "Parabolic Stop And Reverse System"
CiStdDev	Индикатор "Standard Deviation"
CiDEMA	Индикатор "Double Exponential Moving Average"
CiTEMA	Индикатор "Triple Exponential Moving Average"
CiFrAMA	Индикатор "Fractal Adaptive Moving Average"
CiAMA	Индикатор "Adaptive Moving Average"
CiVIDyA	Индикатор "Variable Index Dynamic Average"

© 2000-2025, MetaQuotes Ltd.

Класс CiADX

Класс CiADX является классом для работы с техническим индикатором "Average Directional Index".

Описание

Класс CiADX обеспечивает создание, настройку и доступ к данным индикатора "Average Directional Index".

Декларация

```
class CiADX: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADX

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера основной линии
Plus	Получает данные буфера линии +DI
Minus	Получает данные буфера линии -DI
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,        // символ
   ENUM_TIMEFRAMES  period,        // период
   int              ma_period      // период усреднения
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера основной линии по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера основной линии.

Возвращаемое значение

Элемент буфера основной линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Plus

Получает элемент буфера линии +DI по указанному индексу.

```
double  Plus(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии +DI.

Возвращаемое значение

Элемент буфера линии +DI по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Minus

Получает элемент буфера линии -DI по указанному индексу.

```
double  Minus(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии -DI.

Возвращаемое значение

Элемент буфера линии -DI по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiADX - IND_ADX).

© 2000-2025, MetaQuotes Ltd.

Класс CiADXWilder

Класс CiADXWilder является классом для работы с техническим индикатором "Average Directional Index by Welles Wilder".

Описание

Класс CiADXWilder обеспечивает создание, настройку и доступ к данным индикатора "Average Directional Index by Welles Wilder".

Декларация

```
class CiADXWilder: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADXWilder

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера основной линии
Plus	Получает данные буфера линии +DI
Minus	Получает данные буфера линии -DI
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period       // период усреднения
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера основной линии по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера основной линии.

Возвращаемое значение

Элемент буфера основной линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Plus

Получает элемент буфера линии +DI по указанному индексу.

```
double  Plus(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии +DI.

Возвращаемое значение

Элемент буфера линии +DI по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Minus

Получает элемент буфера линии -DI по указанному индексу.

```
double  Minus(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии -DI.

Возвращаемое значение

Элемент буфера линии -DI по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiADXWilder - IND_ADXW).

© 2000-2025, MetaQuotes Ltd.

Класс CiBands

Класс CiBands является классом для работы с техническим индикатором "Bollinger Bands®".

Описание

Класс CiBands обеспечивает создание, настройку и доступ к данным индикатора "Bollinger Bands".

Декларация

```
class CiBands: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBands

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
MaShift	Получает смещение по оси цен
Deviation	Получает значение девиации
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Base	Получает данные буфера основной линии
Upper	Получает данные буфера верхней линии
Lower	Получает данные буфера нижней линии
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaShift

Получает смещение по оси цен.

```
int  MaShift() const
```

Возвращаемое значение

Смещение по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Deviation

Получает значение девиации.

```
double  Deviation() const
```

Возвращаемое значение

Значение девиации, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period,      // период усреднения
   int               ma_shift,       // смещение
   double            deviation,      // девиация
   int               applied         // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

ma_shift

[in] Смещение индикатора по оси цен.

deviation

[in] Девиация индикатора.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Base

Получает элемент буфера основной линии по указанному индексу.

```
double  Base(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера основной линии.

Возвращаемое значение

Элемент буфера основной линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Upper

Получает элемент буфера верхней линии по указанному индексу.

```
double  Upper(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера верхней линии.

Возвращаемое значение

Элемент буфера верхней линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Lower

Получает элемент буфера нижней линии по указанному индексу.

```
double  Lower(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера нижней линии.

Возвращаемое значение

Элемент буфера нижней линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiBands - IND_BANDS).

© 2000-2025, MetaQuotes Ltd.

Класс CiEnvelopes

Класс CiEnvelopes является классом для работы с техническим индикатором "Envelopes".

Описание

Класс CiEnvelopes обеспечивает создание, настройку и доступ к данным индикатора "Envelopes".

Декларация

```
class CiEnvelopes: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiEnvelopes

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
MaShift	Получает смещение по оси цен
MaMethod	Получает метод усреднения
Deviation	Получает значение девиации
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Upper	Получает данные буфера верхней линии
Lower	Получает данные буфера нижней линии
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaShift

Получает смещение по оси цен.

```
int  MaShift() const
```

Возвращаемое значение

Смещение по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Получает метод усреднения.

```
ENUM_MA_METHOD  MaMethod() const
```

Возвращаемое значение

Метод усреднения, назначенный при создании индикатора (значение перечисления ENUM_MA_METHOD).

© 2000-2025, MetaQuotes Ltd.

Deviation

Получает значение девиации.

```
double  Deviation() const
```

Возвращаемое значение

Значение девиации, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period,      // период усреднения
   int               ma_shift,       // смещение
   ENUM_MA_METHOD    ma_method,      // метод усреднения
   int               applied,        // тип цены, хэндл
   double            deviation       // девиация
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

ma_shift

[in] Смещение индикатора по оси цен.

ma_method

[in] Метод усреднения индикатора (значение перечисления ENUM_MA_METHOD).

applied

[in] Объект (тип цены, хэндл) применения индикатора.

deviation

[in] Девиация индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Upper

Получает элемент буфера верхней линии по указанному индексу.

```
double  Upper(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера верхней линии.

Возвращаемое значение

Элемент буфера верхней линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Lower

Получает элемент буфера нижней линии по указанному индексу.

```
double  Lower(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера нижней линии.

Возвращаемое значение

Элемент буфера нижней линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiEnvelopes - IND_ENVELOPES).

© 2000-2025, MetaQuotes Ltd.

Класс CiIchimoku

Класс CiIchimoku является классом для работы с техническим индикатором "Ichimoku Kinko Hyo".

Описание

Класс CiIchimoku обеспечивает создание, настройку и доступ к данным индикатора "Ichimoku Kinko Hyo".

Декларация

```
class CiIchimoku: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiIchimoku

Методы класса по группам

Атрибуты	
TenkanSenPeriod	Получает период TenkanSen
KijunSenPeriod	Получает период KijunSen
SenkouSpanBPeriod	Получает период SenkouSpanB
Создание	
Create	Создает индикатор
Доступ к данным	
TenkanSen	Получает данные буфера линии TenkanSen
KijunSen	Получает данные буфера линии KijunSen
SenkouSpanA	Получает данные буфера линии SenkouSpanA
SenkouSpanB	Получает данные буфера линии SenkouSpanB
ChinkouSpan	Получает данные буфера линии ChinkouSpan
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

TenkanSenPeriod

Получает период TenkanSen.

```
int  TenkanSenPeriod() const
```

Возвращаемое значение

Период TenkanSen, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

KijunSenPeriod

Получает период KijunSen.

```
int  KijunSenPeriod() const
```

Возвращаемое значение

Период KijunSen, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

SenkouSpanBPeriod

Получает период SenkouSpanB.

```
int  SenkouSpanBPeriod() const
```

Возвращаемое значение

Период SenkouSpanB, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,           // символ
   ENUM_TIMEFRAMES  period,           // период
   int              tenkan_sen,       // период TenkanSen
   int              kijun_sen,        // период KijunSen
   int              senkou_span_b     // период SenkouSpanB
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

tenkan_sen

[in] Период TenkanSen индикатора.

kijun_sen

[in] Период KijunSen индикатора.

senkou_span_b

[in] Период SenkouSpanB индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

TenkanSen

Получает элемент буфера линии TenkanSen по указанному индексу.

```
double  TenkanSen(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии TenkanSen.

Возвращаемое значение

Элемент буфера линии TenkanSen по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

KijunSen

Получает элемент буфера линии KijunSen по указанному индексу.

```
double  KijunSen(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии KijunSen.

Возвращаемое значение

Элемент буфера линии KijunSen по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

SenkouSpanA

Получает элемент буфера линии SenkouSpanA по указанному индексу.

```
double  SenkouSpanA(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии SenkouSpanA.

Возвращаемое значение

Элемент буфера линии SenkouSpanA по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

SenkouSpanB

Получает элемент буфера линии SenkouSpanB по указанному индексу.

```
double  SenkouSpanB(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии SenkouSpanB.

Возвращаемое значение

Элемент буфера линии SenkouSpanB по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

ChinkouSpan

Получает элемент буфера линии ChinkouSpan по указанному индексу.

```
double  ChinkouSpan(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии ChinkouSpan.

Возвращаемое значение

Элемент буфера линии ChinkouSpan по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для Cilchimoku - IND_ICHIMOKU).

© 2000-2025, MetaQuotes Ltd.

Класс CiMA

Класс CiMA является классом для работы с техническим индикатором "Moving Average".

Описание

Класс CiMA обеспечивает создание, настройку и доступ к данным индикатора "Moving Average".

Декларация

```
class CiMA: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMA

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
MaShift	Получает смещение по оси цен
MaMethod	Получает метод усреднения
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaShift

Получает смещение по оси цен.

```
int  MaShift() const
```

Возвращаемое значение

Смещение по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Получает метод усреднения.

```
ENUM_MA_METHOD  MaMethod() const
```

Возвращаемое значение

Метод усреднения, назначенный при создании индикатора (значение перечисления ENUM_MA_METHOD).

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           string,        // символ
   ENUM_TIMEFRAMES  period,        // период
   int              ma_period,     // период усреднения
   int              ma_shift,      // смещение
   ENUM_MA_METHOD   ma_method,     // метод усреднения
   int              applied        // тип цены, хэндл
   )
```

Параметры

string

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

ma_shift

[in] Смещение индикатора по оси цен.

ma_method

[in] Метод усреднения индикатора (значение перечисления ENUM_MA_METHOD).

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiMA - IND_MA).

© 2000-2025, MetaQuotes Ltd.

Класс CiSAR

Класс CiSAR является классом для работы с техническим индикатором "Parabolic Stop And Reverse System".

Описание

Класс CiSAR обеспечивает создание, настройку и доступ к данным индикатора "Parabolic Stop And Reverse System".

Декларация

```
class CiSAR: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiSAR

Методы класса по группам

Атрибуты	
SarStep	Получает шаг изменения цены
Maximum	Получает максимальное значение шага
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

SarStep

Получает шаг изменения цены.

```
double  SarStep() const
```

Возвращаемое значение

Шаг изменения цены, указанный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Maximum

Получает максимальное значение шага.

```
double  Maximum() const
```

Возвращаемое значение

Максимальное значение шага, указанное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,       // символ
   ENUM_TIMEFRAMES   period,       // период
   double            step,         // шаг
   double            maximum       // коэффициент
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

step

[in] Шаг увеличения скорости.

maximum

[in] Коэффициент следования за ценой.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiSAR - IND_SAR).

© 2000-2025, MetaQuotes Ltd.

Класс CiStdDev

Класс CiStdDev является классом для работы с техническим индикатором "Standard Deviation".

Описание

Класс CiStdDev обеспечивает создание, настройку и доступ к данным индикатора "Standard Deviation".

Декларация

```
class CiStdDev: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiStdDev

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
MaShift	Получает смещение по оси цен
MaMethod	Получает метод усреднения
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaShift

Получает смещение по оси цен.

```
int  MaShift() const
```

Возвращаемое значение

Смещение по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Получает метод усреднения.

```
ENUM_MA_METHOD  MaMethod() const
```

Возвращаемое значение

Метод усреднения, назначенный при создании индикатора (значение перечисления ENUM_MA_METHOD).

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // цвет// символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period,      // период усреднения
   int               ma_shift,       // смещение
   ENUM_MA_METHOD    ma_method,      // метод усреднения
   int               applied         // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

ma_shift

[in] Смещение индикатора по оси цен.

ma_method

[in] Метод усреднения индикатора (значение перечисления ENUM_MA_METHOD).

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiStdDev - IND_STDDEV).

© 2000-2025, MetaQuotes Ltd.

Класс CiDEMA

Класс CiDEMA является классом для работы с техническим индикатором "Double Exponential Moving Average".

Описание

Класс CiDEMA обеспечивает создание, настройку и доступ к данным индикатора "Double Exponential Moving Average".

Декларация

```
class CiDEMA: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiDEMA

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
IndShift	Получает смещение по оси цен
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

IndShift

Получает смещение по оси цен.

```
int  IndShift() const
```

Возвращаемое значение

Смещение по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           string,        // символ
   ENUM_TIMEFRAMES  period,        // период
   int              ma_period,     // период усреднения
   int              ind_shift,     // смещение
   int              applied        // тип цены, хэндл
   )
```

Параметры

string

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

ind_shift

[in] Смещение индикатора по оси цен.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiDEMA - IND_DEMA).

© 2000-2025, MetaQuotes Ltd.

Класс CiTEMA

Класс CiTEMA является классом для работы с техническим индикатором " Triple Exponential Moving Average".

Описание

Класс CiTEMA обеспечивает создание, настройку и доступ к данным индикатора " Triple Exponential Moving Average".

Декларация

```
class CiTEMA: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTEMA

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
IndShift	Получает смещение по оси цен
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

IndShift

Получает смещение по оси цен.

```
int  IndShift() const
```

Возвращаемое значение

Смещение по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period,      // период усреднения
   int               ma_shift,       // смещение
   int               applied         // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

ma_shift

[in] Смещение индикатора по оси цен.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiTEMA - IND_TEMA).

© 2000-2025, MetaQuotes Ltd.

Класс CiFrAMA

Класс CiFrAMA является классом для работы с техническим индикатором " Fractal Adaptive Moving Average".

Описание

Класс CiFrAMA обеспечивает создание, настройку и доступ к данным индикатора " Fractal Adaptive Moving Average".

Декларация

```
class CiFrAMA: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiFrAMA

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
IndShift	Получает смещение по оси цен
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

IndShift

Получает смещение по оси цен.

```
int  IndShift() const
```

Возвращаемое значение

Смещение по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period,      // период усреднения
   int               ma_shift,       // смещение
   int               applied         // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

ma_shift

[in] Смещение индикатора по оси цен.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiFrAMA- IND_FRAMA).

© 2000-2025, MetaQuotes Ltd.

Класс CiAMA

Класс CiAMA является классом для работы с техническим индикатором "Adaptive Moving Average".

Описание

Класс CiAMA обеспечивает создание, настройку и доступ к данным индикатора "Adaptive Moving Average".

Декларация

```
class CiAMA: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAMA

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
FastEmaPeriod	Получает период усреднения быстрой EMA
SlowEmaPeriod	Получает период усреднения медленной EMA
IndShift	Получает смещение по оси цен
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

FastEmaPeriod

Получает период усреднения быстрой EMA.

```
int  FastEmaPeriod() const
```

Возвращаемое значение

Период усреднения быстрой EMA, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

SlowEmaPeriod

Получает период усреднения медленной EMA.

```
int  SlowEmaPeriod() const
```

Возвращаемое значение

Период усреднения медленной EMA, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

IndShift

Получает смещение по оси цен.

```
int  IndShift() const
```

Возвращаемое значение

Смещение по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           string,              // символ
   ENUM_TIMEFRAMES  period,              // период
   int              ma_period,           // период усреднения
   int              fast_ema_period,     // период быстрой EMA
   int              slow_ema_period,     // период медленной EMA
   int              ind_shift,           // смещение
   int              applied              // тип цены, хэндл
   )
```

Параметры

string

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

fast_ema_period

[in] Период усреднения быстрой EMA индикатора.

slow_ema_period

[in] Период усреднения медленной EMA индикатора.

ind_shift

[in] Смещение индикатора по оси цен.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiAMA - IND_AMA).

© 2000-2025, MetaQuotes Ltd.

Класс CiVIDyA

Класс CiVIDyA является классом для работы с техническим индикатором "Variable Index Dynamic Average".

Описание

Класс CiVIDyA обеспечивает создание, настройку и доступ к данным индикатора "Variable Index Dynamic Average".

Декларация

```
class CiVIDyA: public CIndicator
```

Заголовок

```
#include <Indicators\Trend.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVIDyA

Методы класса по группам

Атрибуты	
CmoPeriod	Получает период Momentum
EmaPeriod	Получает период сглаживания
IndShift	Получает смещение по оси цен
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

CmoPeriod

Закрывает график привязанный к экземпляру класса.

```
int  CmoPeriod() const
```

© 2000-2025, MetaQuotes Ltd.

EmaPeriod

Получает имя символа графика.

```
int  EmaPeriod() const
```

Возвращаемое значение

Имя символа графика, привязанного к экземпляру класса. Если нет привязанного графика, возвращается "".

© 2000-2025, MetaQuotes Ltd.

IndShift

Получает смещение по оси цен.

```
int  IndShift() const
```

Возвращаемое значение

Смещение по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,          // символ
   ENUM_TIMEFRAMES  period,          // период
   int              cmo_period,      // период Momentum
   int              ema_period,      // период сглаживания
   int              ind_shift,       // смещение
   int              applied          // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

cmo_period

[in] Период Momentum индикатора.

ema_period

[in] Период сглаживания индикатора.

ind_shift

[in] Смещение индикатора по оси цен.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiVIDyA - IND_VIDYA).

© 2000-2025, MetaQuotes Ltd.

Группа классов "Осцилляторы"

Этот раздел содержит детали работы с группой классов "Осцилляторы" и описание соответствующих компонентов стандартной библиотеки MQL5.

Класс/группа	Описание
CiATR	Осциллятор "Average True Range"
CiBearsPower	Осциллятор "Bears Power"
CiBullsPower	Осциллятор "Bulls Power"
CiCCI	Осциллятор "Commodity Channel Index"
CiChaikin	Осциллятор "Chaikin Oscillator"
CiDeMarker	Осциллятор "DeMarker"
CiForce	Осциллятор "Force Index"
CiMACD	Осциллятор "Moving Averages Convergence-Divergence"
CiMomentum	Осциллятор "Momentum"
CiOsMA	Осциллятор "Moving Average of Oscillator (MACD histogram)"
CiRSI	Осциллятор "Relative Strength Index"
CiRVI	Осциллятор "Relative Vigor Index"
CiStochastic	Осциллятор "Stochastic Oscillator"
CiWPR	Осциллятор "Williams' Percent Range"
CiTriX	Осциллятор "Triple Exponential Moving Averages Oscillator"

© 2000-2025, MetaQuotes Ltd.

Класс CiATR

Класс CiATR является классом для работы с техническим индикатором "Average True Range".

Описание

Класс CiATR обеспечивает создание, настройку и доступ к данным индикатора "Average True Range".

Декларация

```
class CiATR: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiATR

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period       // период усреднения
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiATR - IND_ATR).

© 2000-2025, MetaQuotes Ltd.

Класс CiBearsPower

Класс CiBearsPower является классом для работы с техническим индикатором "Bears Power".

Описание

Класс CiBearsPower обеспечивает создание, настройку и доступ к данным индикатора "Bears Power".

Декларация

```
class CiBearsPower: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBearsPower

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period       // период усреднения
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiBearsPower - IND_BEARS).

© 2000-2025, MetaQuotes Ltd.

Класс CiBullsPower

Класс CiBullsPower является классом для работы с техническим индикатором "Bulls Power".

Описание

Класс CiBullsPower обеспечивает создание, настройку и доступ к данным индикатора "Bulls Power".

Декларация

```
class CiBullsPower: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBullsPower

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period       // период усреднения
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiBullsPower - IND_BULLS).

© 2000-2025, MetaQuotes Ltd.

Класс CiCCI

Класс CiCCI является классом для работы с техническим индикатором "Commodity Channel Index".

Описание

Класс CiCCI обеспечивает создание, настройку и доступ к данным индикатора "Commodity Channel Index".

Декларация

```
class CiCCI: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiCCI

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period,      // период усреднения
   int               applied         // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiCCI - IND_CCI).

© 2000-2025, MetaQuotes Ltd.

Класс CiChaikin

Класс CiChaikin является классом для работы с техническим индикатором "Chaikin Oscillator".

Описание

Класс CiChaikin обеспечивает создание, настройку и доступ к данным индикатора "Chaikin Oscillator".

Декларация

```
class CiChaikin: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiChaikin

Методы класса по группам

Атрибуты	
FastMaPeriod	Получает период усреднения быстрой MA
SlowMaPeriod	Получает период усреднения медленной MA
MaMethod	Получает период усреднения
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

FastMaPeriod

Получает период усреднения быстрой EMA.

```
int  FastMaPeriod() const
```

Возвращаемое значение

Период усреднения быстрой EMA, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

SlowMaPeriod

Получает период усреднения медленной EMA.

```
int  SlowMaPeriod() const
```

Возвращаемое значение

Период усреднения медленной EMA, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Получает метод усреднения.

```
ENUM_MA_METHOD  MaMethod() const
```

Возвращаемое значение

Метод усреднения, назначенный при создании индикатора (значение перечисления ENUM_MA_METHOD).

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип объемов) применения.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Возвращаемое значение

Объект (тип объемов) применения, назначенный при создании индикатора (значение перечисления ENUM_APPLIED_VOLUME).

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string               symbol,              // символ
   ENUM_TIMEFRAMES      period,              // период
   int                  fast_ma_period,      // период быстрой EMA
   int                  slow_ma_period,      // период медленной EMA
   ENUM_MA_METHOD       ma_method,           // метод усреднения
   ENUM_APPLIED_VOLUME  applied              // тип объемов
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

fast_ma_period

[in] Период усреднения быстрой EMA индикатора.

slow_ma_period

[in] Период усреднения медленной EMA индикатора.

ma_method

[in] Метод усреднения индикатора (значение перечисления ENUM_MA_METHOD).

applied

[in] Объект (тип объемов) применения индикатора (значение перечисления ENUM_APPLIED_VOLUME).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiChaikin - IND_CHAIKIN).

© 2000-2025, MetaQuotes Ltd.

Класс CiDeMarker

Класс CiDeMarker является классом для работы с техническим индикатором "DeMarker".

Описание

Класс CiDeMarker обеспечивает создание, настройку и доступ к данным индикатора "DeMarker".

Декларация

```
class CiDeMarker: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiDeMarker

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period       // период усреднения
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiDeMarker - IND_DEMARKER).

© 2000-2025, MetaQuotes Ltd.

Класс CiForce

Класс CiForce является классом для работы с техническим индикатором "Force Index".

Описание

Класс CiForce обеспечивает создание, настройку и доступ к данным индикатора "Force Index".

Декларация

```
class CiForce: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiForce

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
MaMethod	Получает метод усреднения
Applied	Получает объект (тип объемов) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Получает метод усреднения.

```
ENUM_MA_METHOD  MaMethod() const
```

Возвращаемое значение

Метод усреднения, назначенный при создании индикатора (значение перечисления ENUM_MA_METHOD).

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип объемов) применения.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Возвращаемое значение

Объект (тип объемов) применения, назначенный при создании индикатора (значение перечисления ENUM_APPLIED_VOLUME).

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string                symbol,         // символ
   ENUM_TIMEFRAMES       period,         // период
   int                   ma_period,      // период усреднения
   ENUM_MA_METHOD        ma_method,      // метод усреднения
   ENUM_APPLIED_VOLUME   applied         // тип объемов
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

ma_method

[in] Метод усреднения индикатора (значение перечисления ENUM_MA_METHOD).

applied

[in] Объект (тип объемов) применения индикатора (значение перечисления ENUM_APPLIED_VOLUME).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiForce - IND_FORCE).

© 2000-2025, MetaQuotes Ltd.

Класс CiMACD

Класс CiMACD является классом для работы с техническим индикатором "Moving Averages Convergence-Divergence".

Описание

Класс CiMACD обеспечивает создание, настройку и доступ к данным индикатора "Moving Averages Convergence-Divergence".

Декларация

```
class CiMACD: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMACD

Методы класса по группам

Атрибуты	
FastEmaPeriod	Получает период усреднения быстрой EMA
SlowEmaPeriod	Получает период усреднения медленной EMA
SignalPeriod	Получает период усреднения сигнальной линии
Applied	Получает объект (тип объемов) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера основной линии
Signal	Получает данные буфера сигнальной линии
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

FastEmaPeriod

Получает период усреднения быстрой EMA.

```
int  FastEmaPeriod() const
```

Возвращаемое значение

Период усреднения быстрой EMA, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

SlowEmaPeriod

Получает период усреднения медленной EMA.

```
int  SlowEmaPeriod() const
```

Возвращаемое значение

Период усреднения медленной EMA, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

SignalPeriod

Получает период усреднения сигнальной линии.

```
int  SignalPeriod() const
```

Возвращаемое значение

Период усреднения сигнальной линии, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,              // символ
   ENUM_TIMEFRAMES  period,              // период
   int              fast_ema_period,     // период быстрой EMA
   int              slow_ema_period,     // период медленной EMA
   int              signal_period,       // период сигнала
   int              applied              // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

fast_ema_period

[in] Период усреднения быстрой EMA индикатора.

slow_ema_period

[in] Период усреднения медленной EMA индикатора.

signal_period

[in] Период усреднения сигнальной линии индикатора.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера основной линии по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера основной линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Signal

Получает элемент буфера сигнальной линии по указанному индексу.

```
double  Signal(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера сигнальной линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiMACD - IND_MACD).

© 2000-2025, MetaQuotes Ltd.

Класс CiMomentum

Класс CiMomentum является классом для работы с техническим индикатором "Momentum".

Описание

Класс CiMomentum обеспечивает создание, настройку и доступ к данным индикатора "Momentum".

Декларация

```
class CiMomentum: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMomentum

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Applied	Получает объект (тип объемов) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period,      // период усреднения
   int               applied         // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiMomentum - IND_MOMENTUM).

© 2000-2025, MetaQuotes Ltd.

Класс CiOsMA

Класс CiOsMA является классом для работы с техническим индикатором "Moving Average of Oscillator (MACD histogram)".

Описание

Класс CiOsMA обеспечивает создание, настройку и доступ к данным индикатора "Moving Average of Oscillator (MACD histogram)".

Декларация

```
class CiOsMA: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOsMA

Методы класса по группам

Атрибуты	
FastEmaPeriod	Получает период усреднения быстрой EMA
SlowEmaPeriod	Получает период усреднения медленной EMA
SignalPeriod	Получает период усреднения сигнальной линии
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

FastEmaPeriod

Получает период усреднения быстрой EMA.

```
int  FastEmaPeriod() const
```

Возвращаемое значение

Период усреднения быстрой EMA, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

SlowEmaPeriod

Получает период усреднения медленной EMA.

```
int  SlowEmaPeriod() const
```

Возвращаемое значение

Период усреднения медленной EMA, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

SignalPeriod

Получает период усреднения сигнальной линии.

```
int  SignalPeriod() const
```

Возвращаемое значение

Период усреднения сигнальной линии, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,              // символ
   ENUM_TIMEFRAMES  period,              // период
   int              fast_ema_period,     // период быстрой EMA
   int              slow_ema_period,     // период медленной EMA
   int              signal_period,       // период сигнала
   int              applied              // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

fast_ema_period

[in] Период усреднения быстрой EMA индикатора.

slow_ema_period

[in] Период усреднения медленной EMA индикатора.

signal_period

[in] Период усреднения сигнальной линии индикатора.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiOsMA - IND_OSMA).

© 2000-2025, MetaQuotes Ltd.

Класс CiRSI

Класс CiRSI является классом для работы с техническим индикатором "Relative Strength Index".

Описание

Класс CiRSI обеспечивает создание, настройку и доступ к данным индикатора "Relative Strength Index".

Декларация

```
class CiRSI: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRSI

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period,      // период усреднения
   int               applied         // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiRSI - IND_RSI).

© 2000-2025, MetaQuotes Ltd.

Класс CiRVI

Класс CiRVI является классом для работы с техническим индикатором "Relative Vigor Index".

Описание

Класс CiRVI обеспечивает создание, настройку и доступ к данным индикатора "Relative Vigor Index".

Декларация

```
class CiRVI: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRVI

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера основной линии
Signal	Получает данные буфера сигнальной линии
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,        // символ
   ENUM_TIMEFRAMES  period,        // период
   int              ma_period      // период усреднения
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера основной линии по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера основной линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Signal

Получает элемент буфера сигнальной линии по указанному индексу.

```
double  Signal(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера сигнальной линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiRVI - IND_RVI).

© 2000-2025, MetaQuotes Ltd.

Класс CiStochastic

Класс CiStochastic является классом для работы с техническим индикатором "Stochastic Oscillator".

Описание

Класс CiStochastic обеспечивает создание, настройку и доступ к данным индикатора "Stochastic Oscillator".

Декларация

```
class CiStochastic: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiStochastic

Методы класса по группам

Атрибуты	
Kperiod	Получает период усреднения %K
Dperiod	Получает период усреднения %D
Slowing	Получает период замедления
MaMethod	Получает метод усреднения
PriceField	Получает объект (Low/High или Close/Close) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера основной линии
Signal	Получает данные буфера сигнальной линии
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Kperiod

Получает период усреднения %K.

```
int  Kperiod() const
```

Возвращаемое значение

Период усреднения %K, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Dperiod

Получает период усреднения %D.

```
int  Dperiod() const
```

Возвращаемое значение

Период усреднения %D, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Slowing

Получает период замедления.

```
int  Slowing() const
```

Возвращаемое значение

Период замедления, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Получает метод усреднения.

```
ENUM_MA_METHOD  MaMethod() const
```

Возвращаемое значение

Метод усреднения, назначенный при создании индикатора (значение перечисления ENUM_MA_METHOD).

© 2000-2025, MetaQuotes Ltd.

PriceField

Получает объект (Low/High или Close/Close) применения.

```
ENUM_STO_PRICE  PriceField() const
```

Возвращаемое значение

Объект (Low/High или Close/Close) применения, назначенный при создании индикатора (значение перечисления ENUM_STO_PRICE).

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,          // символ
   ENUM_TIMEFRAMES  period,          // период
   int              Kperiod,         // период %K
   int              Dperiod,         // период %D
   int              slowing,         // период замедления
   ENUM_MA_METHOD   ma_method,       // метод усреднения
   ENUM_STO_PRICE   price_field      // применение
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

Kperiod

[in] Период усреднения %K индикатора.

Dperiod

[in] Период усреднения %D индикатора.

slowing

[in] Период замедления индикатора.

ma_method

[in] Метод усреднения индикатора (значение перечисления ENUM_MA_METHOD).

price_field

[in] Объект (Low/High или Close/Close) применения индикатора (значение перечисления ENUM_STO_PRICE).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера основной линии по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера основной линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Signal

Получает элемент буфера сигнальной линии по указанному индексу.

```
double  Signal(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера сигнальной линии по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiStochastic - IND_STOCHASTIC).

© 2000-2025, MetaQuotes Ltd.

Класс CiTriX

Класс CiTriX является классом для работы с техническим индикатором "Triple Exponential Moving Averages Oscillator".

Описание

Класс CiTriX обеспечивает создание, настройку и доступ к данным индикатора "Triple Exponential Moving Averages Oscillator".

Декларация

```
class CiTriX: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTriX

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,         // символ
   ENUM_TIMEFRAMES   period,         // период
   int               ma_period,      // период усреднения
   int               applied         // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiTriX - IND_TRIX).

© 2000-2025, MetaQuotes Ltd.

Класс CiWPR

Класс CiWPR является классом для работы с техническим индикатором "Williams' Percent Range".

Описание

Класс CiWPR обеспечивает создание, настройку и доступ к данным индикатора "Williams' Percent Range".

Декларация

```
class CiWPR: public CIndicator
```

Заголовок

```
#include <Indicators\Oscilators.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiWPR

Методы класса по группам

Атрибуты	
CalcPeriod	Получает период расчета
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

CalcPeriod

Получает период расчета.

```
int  CalcPeriod() const
```

Возвращаемое значение

Период расчета, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,          // символ
   ENUM_TIMEFRAMES  period,          // период
   int              calc_period      // период расчета
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

calc_period

[in] Период расчета индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiWPR - IND_WPR).

© 2000-2025, MetaQuotes Ltd.

Группа технических индикаторов "Объемы"

Этот раздел содержит детали работы с группой классов технических индикаторов "Объемы" и описание соответствующих компонентов стандартной библиотеки MQL5.

Класс/группа	Описание
CiAD	Индикатор "Accumulation/Distribution"
CiMFI	Индикатор "Money Flow Index"
CiOBV	Индикатор "On Balance Volume"
CiVolumes	Индикатор "Volumes"

© 2000-2025, MetaQuotes Ltd.

Класс CiAD

Класс CiAD является классом для работы с техническим индикатором "Accumulation/Distribution".

Описание

Класс CiAD обеспечивает создание, настройку и доступ к данным индикатора "Accumulation/Distribution".

Декларация

```
class CiAD: public CIndicator
```

Заголовок

```
#include <Indicators\Volumes.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAD

Методы класса по группам

Атрибуты	
Applied	Получает период расчета
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип объемов) применения.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Возвращаемое значение

Объект (тип объемов) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string                symbol,      // символ
   ENUM_TIMEFRAMES       period,      // период
   ENUM_APPLIED_VOLUME   applied      // тип объемов
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

applied

[in] Объект (тип объемов) применения индикатора (ENUM_APPLIED_VOLUME).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiAD - IND_AD).

© 2000-2025, MetaQuotes Ltd.

Класс CiMFI

Класс CiMFI является классом для работы с техническим индикатором "Money Flow Index".

Описание

Класс CiMFI обеспечивает создание, настройку и доступ к данным индикатора "Money Flow Index".

Декларация

```
class CiMFI: public CIndicator
```

Заголовок

```
#include <Indicators\Volumes.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMFI

Методы класса по группам

Атрибуты	
MaPeriod	Получает период усреднения
Applied	Получает объект (тип объемов) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

Получает период усреднения.

```
int  MaPeriod() const
```

Возвращаемое значение

Период усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип объемов) применения.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Возвращаемое значение

Объект (тип объемов) применения, назначенный при создании индикатора (значение перечисления ENUM_APPLIED_VOLUME).

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string                symbol,         // символ
   ENUM_TIMEFRAMES       period,         // период
   int                   ma_period,      // период усреднения
   ENUM_APPLIED_VOLUME   applied         // тип объемов
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

ma_period

[in] Период усреднения индикатора.

applied

[in] Объект (тип объемов) применения индикатора (значение перечисления ENUM_APPLIED_VOLUME).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiMFI - IND_MFI).

© 2000-2025, MetaQuotes Ltd.

Класс CiMFI

Класс CiOBV является классом для работы с техническим индикатором "On Balance Volume".

Описание

Класс CiOBV обеспечивает создание, настройку и доступ к данным индикатора "On Balance Volume".

Декларация

```
class CiOBV: public CIndicator
```

Заголовок

```
#include <Indicators\Volumes.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiOBV

Методы класса по группам

Атрибуты	
Applied	Получает объект (тип объемов) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип объемов) применения.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Возвращаемое значение

Объект (тип объемов) применения, назначенный при создании индикатора (значение перечисления ENUM_APPLIED_VOLUME).

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string                symbol,      // символ
   ENUM_TIMEFRAMES       period,      // период
   ENUM_APPLIED_VOLUME   applied      // тип объемов
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

applied

[in] Объект (тип объемов) применения индикатора (значение перечисления ENUM_APPLIED_VOLUME).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiOBV - IND_OBV).

© 2000-2025, MetaQuotes Ltd.

Класс CiVolumes

Класс CiVolumes является классом для работы с техническим индикатором "Volumes".

Описание

Класс CiVolumes обеспечивает создание, настройку и доступ к данным индикатора "Volumes".

Декларация

```
class CiVolumes: public CIndicator
```

Заголовок

```
#include <Indicators\Volumes.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiVolumes

Методы класса по группам

Атрибуты	
Applied	Получает объект (тип объемов) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип объемов) применения.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Возвращаемое значение

Объект (тип объемов) применения, назначенный при создании индикатора (значение перечисления ENUM_APPLIED_VOLUME).

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string                symbol,      // символ
   ENUM_TIMEFRAMES       period,      // период
   ENUM_APPLIED_VOLUME   applied      // тип объемов
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение пересечения ENUM_TIMEFRAMES).

applied

[in] Объект (тип объемов) применения индикатора (значение перечисления ENUM_APPLIED_VOLUME).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiVolumes - IND_VOLUMES).

© 2000-2025, MetaQuotes Ltd.

Группа технических индикаторов Билла Вильямса

Этот раздел содержит детали работы с группой классов технических индикаторов Билла Вильямса и описание соответствующих компонентов стандартной библиотеки MQL5.

Класс/группа	Описание
CiAC	Индикатор "Accelerator Oscillator"
CiAlligator	Индикатор "Alligator"
CiAO	Индикатор "Awesome Oscillator"
CiFractals	Индикатор "Fractals"
CiGator	Индикатор "Gator Oscillator"
CiBWMFI	Индикатор "Market Facilitation Index"

© 2000-2025, MetaQuotes Ltd.

Класс CiAC

Класс CiAC является классом для работы с техническим индикатором "Accelerator Oscillator".

Описание

Класс CiAC обеспечивает создание, настройку и доступ к данным индикатора "Accelerator Oscillator".

Декларация

```
class CiAC: public CIndicator
```

Заголовок

```
#include <Indicators\BillWilliams.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAC

Методы класса по группам

Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string               symbol,      // символ
   ENUM_TIMEFRAMES      period       // период
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiAC - IND_AC).

© 2000-2025, MetaQuotes Ltd.

Класс CiAlligator

Класс CiAlligator является классом для работы с техническим индикатором "Alligator".

Описание

Класс CiAlligator обеспечивает создание, настройку и доступ к данным индикатора "Alligator".

Декларация

```
class CiAlligator: public CIndicator
```

Заголовок

```
#include <Indicators\BillWilliams.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAlligator

Методы класса по группам

Атрибуты	
JawPeriod	Получает период усреднения "челюстей"
JawShift	Получает смещение "челюстей" по оси цен
TeethPeriod	Получает период усреднения "зубов"
TeethShift	Получает смещение "зубов" по оси цен
LipsPeriod	Получает период усреднения "губ"
LipsShift	Получает смещение "губ" по оси цен
MaMethod	Получает метод усреднения
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Jaw	Получает данные буфера линии "челюстей"
Teeth	Получает данные буфера линии "зубов"

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
Lips	Получает данные буфера линии "губ"
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

JawPeriod

Получает период усреднения линии "челюстей".

```
int  JawPeriod() const
```

Возвращаемое значение

Период усреднения линии "челюстей", назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

JawShift

Получает смещение линии "челюстей" по оси цен.

```
int  JawShift() const
```

Возвращаемое значение

Смещение линии "челюстей" по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

TeethPeriod

Получает период усреднения линии "зубов".

```
int  TeethPeriod() const
```

Возвращаемое значение

Период усреднения линии "зубов", назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

TeethShift

Получает смещение линии "зубов" по оси цен.

```
int  TeethShift() const
```

Возвращаемое значение

Смещение линии "зубов" по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

LipsPeriod

Получает период усреднения линии "губ".

```
int  LipsPeriod() const
```

Возвращаемое значение

Период усреднения линии "губ", назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

LipsShift

Получает смещение линии "губ" по оси цен.

```
int  LipsShift() const
```

Возвращаемое значение

Смещение линии "губ" по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Получает метод усреднения.

```
ENUM_MA_METHOD  MaMethod() const
```

Возвращаемое значение

Метод усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,           // символ
   ENUM_TIMEFRAMES  period,           // период
   int              jaw_period,       // период "челюстей"
   int              jaw_shift,        // смещение "челюстей"
   int              teeth_period,     // период "зубов"
   int              teeth_shift,      // смещение "зубов"
   int              lips_period,      // период "губ"
   int              lips_shift,       // смещение "губ"
   ENUM_MA_METHOD   ma_method,        // метод усреднения
   int              applied           // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

jaw_period

[in] Период усреднения линии "челюстей".

jaw_shift

[in] Смещение линии "челюстей" по оси цен.

teeth_period

[in] Период усреднения линии "зубов".

teeth_shift

[in] Смещение линии "зубов" по оси цен.

lips_period

[in] Период усреднения линии "губ".

lips_shift

[in] Смещение линии "губ" по оси цен.

ma_method

[in] Метод усреднения индикатора (значение перечисления ENUM_MA_METHOD).

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Jaw

Получает элемент буфера линии "челюстей" по указанному индексу.

```
double  Jaw(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии "челюстей".

Возвращаемое значение

Элемент буфера линии "челюстей" по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Teeth

Получает элемент буфера линии "зубов" по указанному индексу.

```
double  Teeth(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии "зубов".

Возвращаемое значение

Элемент буфера линии "зубов" по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Lips

Получает элемент буфера линии "губ" по указанному индексу.

```
double  Lips(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера линии "губ".

Возвращаемое значение

Элемент буфера линии "губ" по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiAlligator - IND_ALLIGATOR).

© 2000-2025, MetaQuotes Ltd.

Класс CiAO

Класс CiAO является классом для работы с техническим индикатором "Awesome Oscillator".

Описание

Класс CiAO обеспечивает создание, настройку и доступ к данным индикатора "Awesome Oscillator".

Декларация

```
class CiAO: public CIndicator
```

Заголовок

```
#include <Indicators\BillWilliams.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAO

Методы класса по группам

Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,      // символ
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiAO - IND_AO).

© 2000-2025, MetaQuotes Ltd.

Класс CiFractals

Класс CiFractals является классом для работы с техническим индикатором "Fractals".

Описание

Класс CiFractals обеспечивает создание, настройку и доступ к данным индикатора "Fractals".

Декларация

```
class CiFractals: public CIndicator
```

Заголовок

```
#include <Indicators\BillWilliams.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiFractals

Методы класса по группам

Создание	
Create	Создает индикатор
Доступ к данным	
Upper	Получает данные верхнего буфера
Lower	Получает данные нижнего буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string           symbol,      // символ
   ENUM_TIMEFRAMES  period       // период
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Upper

Получает элемент верхнего буфера по указанному индексу.

```
double  Upper(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента верхнего буфера.

Возвращаемое значение

Элемент верхнего буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Lower

Получает элемент нижнего буфера по указанному индексу.

```
double  Lower(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента нижнего буфера.

Возвращаемое значение

Элемент нижнего буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiFractals - IND_FRACTALS).

© 2000-2025, MetaQuotes Ltd.

Класс CiGator

Класс CiGator является классом для работы с техническим индикатором "Gator oscillator".

Описание

Класс CiGator обеспечивает создание, настройку и доступ к данным индикатора "Gator oscillator".

Декларация

```
class CiGator: public CIndicator
```

Заголовок

```
#include <Indicators\BillWilliams.mqh>
```

Иерархия наследования

CObject
CArray
CArrayObj
CSeries
CIndicator
CiGator

Методы класса по группам

Атрибуты	
JawPeriod	Получает период усреднения "челюстей"
JawShift	Получает смещение "челюстей" по оси цен
TeethPeriod	Получает период усреднения "зубов"
TeethShift	Получает смещение "зубов" по оси цен
LipsPeriod	Получает период усреднения "губ"
LipsShift	Получает смещение "губ" по оси цен
MaMethod	Получает метод усреднения
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Upper	Получает данные верхнего буфера
Lower	Получает данные нижнего буфера

© 2000-2025, MetaQuotes Ltd.

Атрибуты	
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

JawPeriod

Получает период усреднения линии "челюстей".

```
int  JawPeriod() const
```

Возвращаемое значение

Период усреднения линии "челюстей", назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

JawShift

Получает смещение линии "челюстей" по оси цен.

```
int  JawShift() const
```

Возвращаемое значение

Смещение линии "челюстей" по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

TeethPeriod

Получает период усреднения линии "зубов".

```
int  TeethPeriod() const
```

Возвращаемое значение

Период усреднения линии "зубов", назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

TeethShift

Получает смещение линии "зубов" по оси цен.

```
int  TeethShift() const
```

Возвращаемое значение

Смещение линии "зубов" по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

LipsPeriod

Получает период усреднения линии "губ".

```
int  LipsPeriod() const
```

Возвращаемое значение

Период усреднения линии "губ", назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

LipsShift

Получает смещение линии "губ" по оси цен.

```
int  LipsShift() const
```

Возвращаемое значение

Смещение линии "губ" по оси цен, назначенное при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

MaMethod

Получает метод усреднения.

```
ENUM_MA_METHOD  MaMethod() const
```

Возвращаемое значение

Метод усреднения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип цены, хэндл) применения.

```
int  Applied() const
```

Возвращаемое значение

Объект (тип цены, хэндл) применения, назначенный при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string            symbol,           // символ
   ENUM_TIMEFRAMES   period,           // период
   int               jaw_period,       // период "челюстей"
   int               jaw_shift,        // смещение "челюстей"
   int               teeth_period,     // период "зубов"
   int               teeth_shift,      // смещение "зубов"
   int               lips_period,      // период "губ"
   int               lips_shift,       // смещение "губ"
   ENUM_MA_METHOD    ma_method,        // метод усреднения
   int               applied           // тип цены, хэндл
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

jaw_period

[in] Период усреднения линии "челюстей".

jaw_shift

[in] Смещение линии "челюстей" по оси цен.

teeth_period

[in] Период усреднения линии "зубов".

teeth_shift

[in] Смещение линии "зубов" по оси цен.

lips_period

[in] Период усреднения линии "губ".

lips_shift

[in] Смещение линии "губ" по оси цен.

ma_method

[in] Метод усреднения индикатора (значение перечисления ENUM_MA_METHOD).

applied

[in] Объект (тип цены, хэндл) применения индикатора.

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Upper

Получает элемент верхнего буфера по указанному индексу.

```
double  Upper(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента верхнего буфера.

Возвращаемое значение

Элемент верхнего буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Lower

Получает элемент нижнего буфера по указанному индексу.

```
double  Lower(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента нижнего буфера.

Возвращаемое значение

Элемент нижнего буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiGator - IND_GATOR).

© 2000-2025, MetaQuotes Ltd.

Класс CiBWMFI

Класс CiBWMFI является классом для работы с техническим индикатором "Market Facilitation Index by Bill Williams".

Описание

Класс CiBWMFI обеспечивает создание, настройку и доступ к данным индикатора "Market Facilitation Index by Bill Williams".

Декларация

```
class CiBWMFI: public CIndicator
```

Заголовок

```
#include <Indicators\BillWilliams.mqh>
```

Иерархия наследования

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBWMFI

Методы класса по группам

Атрибуты	
Applied	Получает объект (тип цены, хэндл) применения
Создание	
Create	Создает индикатор
Доступ к данным	
Main	Получает данные буфера
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

Методы унаследованные от CObject

Prev, Prev, Next, Next, Compare

Методы унаследованные от CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Методы унаследованные от CArrayObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

Методы унаследованные от CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Методы унаследованные от CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Applied

Получает объект (тип объемов) применения.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Возвращаемое значение

Объект (тип объемов) применения, назначенный при создании индикатора (значение перечисления ENUM_APPLIED_VOLUME).

© 2000-2025, MetaQuotes Ltd.

Create

Создает индикатор с указанными параметрами. Для обновления и получения значений индикатора используйте Refresh() и GetData().

```
bool  Create(
   string                symbol,      // символ
   ENUM_TIMEFRAMES       period,      // период
   ENUM_APPLIED_VOLUME   applied      // тип объемов
   )
```

Параметры

symbol

[in] Рабочий символ индикатора.

period

[in] Рабочий период индикатора (значение перечисления ENUM_TIMEFRAMES).

applied

[in] Объект (тип объемов) применения индикатора (значение перечисления ENUM_APPLIED_VOLUME).

Возвращаемое значение

true - в случае удачи, false - если не удалось создать индикатор.

© 2000-2025, MetaQuotes Ltd.

Main

Получает элемент буфера по указанному индексу.

```
double  Main(
   int  index      // индекс
   ) const
```

Параметры

index

[in] Индекс элемента буфера.

Возвращаемое значение

Элемент буфера по указанному индексу, либо EMPTY_VALUE если нет корректных данных.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiBWMFI - IND_BWMFI).

© 2000-2025, MetaQuotes Ltd.

Класс CiCustom

Класс CiCustom является классом для работы с техническим индикатором пользовательского типа.

Описание

Класс CiCustom обеспечивает создание, настройку и доступ к данным индикатора пользовательского типа.

Декларация

```
class CiCustom: public CIndicator
```

Заголовок

```
#include <Indicators\Custom.mqh>
```

Методы класса по группам

Атрибуты	
NumBuffers	Устанавливает количество буферов индикатора
NumParams	Получает количество параметров использованных при создании индикатора
ParamType	Получает тип указанного параметра
ParamLong	Получает значение указанного целочисленного параметра
ParamDouble	Получает значение указанного параметра с плавающей точкой
ParamString	Получает значение указанного строкового параметра
Ввод/вывод	
virtual Type	Виртуальный метод идентификации

© 2000-2025, MetaQuotes Ltd.

NumBuffers

Устанавливает количество буферов индикатора.

```
bool  NumBuffers()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось установить нужное количество буферов.

© 2000-2025, MetaQuotes Ltd.

NumParams

Получает количество параметров использованных при создании индикатора.

```
int  NumParams() const
```

Возвращаемое значение

Количество параметров использованных при создании индикатора.

© 2000-2025, MetaQuotes Ltd.

ParamType

Получает тип указанного параметра.

```
ENUM_DATATYPE  ParamType(
   int  index      // номер
   ) const
```

Параметры

index

[in] Номер параметра.

Возвращаемое значение

Получает тип указанного параметра, использованного при создании индикатора.

Примечание

Если номер параметра указан неверно, вернётся WRONG_VALUE.

© 2000-2025, MetaQuotes Ltd.

ParamLong

Получает значение указанного целочисленного параметра.

```
long  ParamLong(
   int  index      // номер
   ) const
```

Параметры

index

[in] Номер параметра.

Возвращаемое значение

Значение указанного целочисленного параметра, использованного при создании индикатора.

Примечание

Если номер параметра указан неверно или тип параметра не целочисленный, вернётся 0.

© 2000-2025, MetaQuotes Ltd.

ParamDouble

Получает значение указанного параметра с плавающей точкой.

```
double  ParamDouble(
   int  index      // номер
   ) const
```

Параметры

index

[in] Номер параметра.

Возвращаемое значение

Значение указанного параметра с плавающей точкой, использованного при создании индикатора.

Примечание

Если номер параметра указан неверно или тип параметра не с плавающей точкой, вернется EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

ParamString

Получает значение указанного строкового параметра.

```
string  ParamString(
   int  index      // номер
   ) const
```

Параметры

index

[in] Номер параметра.

Возвращаемое значение

Значение указанного строкового параметра, использованного при создании индикатора.

Примечание

Если номер параметра указан неверно или тип параметра не строковый, вернется пустая строка.

© 2000-2025, MetaQuotes Ltd.

Type

Виртуальный метод идентификации.

```
virtual int  Type() const
```

Возвращаемое значение

Тип индикатора (для CiCustom - IND_CUSTOM).

© 2000-2025, MetaQuotes Ltd.

Торговые классы

Этот раздел содержит технические детали работы с торговыми классами и описание соответствующих компонентов стандартной библиотеки MQL5.

Использование торговых классов позволит сэкономить время при создании пользовательских программ (экспертов).

Стандартная библиотека MQL5 (в части торговых классов) размещается в рабочем каталоге терминала в папке Include\Trade.

Класс/группа	Описание
CAccountInfo	Класс для работы со свойствами торгового счета
CSymbolInfo	Класс для работы со свойствами торгового инструмента
COrderInfo	Класс для работы со свойствами отложенного ордера
CHistoryOrderInfo	Класс для работы со свойствами "исторического" ордера
CPositionInfo	Класс для работы со свойствами открытой позиции
CDealInfo	Класс для работы со свойствами "исторической" сделки
CTrade	Класс для совершения торговых операций
CTerminalInfo	Класс для доступа к параметрам окружения терминала

© 2000-2025, MetaQuotes Ltd.

Класс CAccountInfo

Класс CAccountInfo является классом для упрощенного доступа к свойствам текущего открытого торгового счета.

Описание

Класс CAccountInfo обеспечивает доступ к свойствам текущего открытого торгового счета.

Декларация

```
class CAccountInfo : public CObject
```

Заголовок

```
#include <Trade\AccountInfo.mqh>
```

Иерархия наследования

CObject

CAccountInfo

Методы класса по группам

Доступ к целочисленным свойствам	
Login	Получает номер счета
TradeMode	Получает режим торговли
TradeModeDescription	Получает режим торговли как строку
Leverage	Получает размер предоставленного кредитного плеча
StopoutMode	Получает режим задания минимального уровня залога
StopoutModeDescription	Получает режим задания минимального уровня залога как строку
TradeAllowed	Получает флаг разрешения торговли
TradeExpert	Получает флаг разрешения автоматизированной торговли
LimitOrders	Получает максимально допустимое количество действующих отложенных ордеров
MarginMode	Получает режим расчета маржи
MarginModeDescription	Получает режим расчета маржи как строку
Доступ к double-свойствам	
Balance	Получает баланс счета
Credit	Получает размер предоставленного кредита

© 2000-2025, MetaQuotes Ltd.

Доступ к целочисленным свойствам	
Profit	Получает значение текущей прибыли по счету
Equity	Получает значение собственных средств на счете
Margin	Получает размер зарезервированных залоговых средств
FreeMargin	Получает размер свободных средств
MarginLevel	Получает уровень залоговых средств
MarginCall	Получает уровень залоговых средств пополнения счета
MarginStopOut	Получает уровень залоговых средств принудительного закрытия
Доступ к текстовым свойствам	
Name	Получает имя клиента
Server	Получает наименование торгового сервера
Currency	Получает наименование валюты депозита
Company	Получает наименование компании, обслуживающей счет
Доступ к функциям API MQL5	
InfoInteger	Получает значение указанного целочисленного свойства счета
InfoDouble	Получает значение указанного double-свойства счета
InfoString	Получает значение указанного текстового свойства счета
Дополнительные методы	
OrderProfitCheck	Вычисляет размер прибыли на основании переданных параметров
MarginCheck	Получает размер маржи, необходимой для торговой операции
FreeMarginCheck	Получает размер свободной маржи, которая останется после торговой операции
MaxLotCheck	Получает максимально возможный объем торговой операции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Login

Получает номер счета.

```
long  Login() const
```

Возвращаемое значение

Номер счета.

© 2000-2025, MetaQuotes Ltd.

TradeMode

Получает режим торговли.

```
ENUM_ACCOUNT_TRADE_MODE  TradeMode() const
```

Возвращаемое значение

Режим торговли из перечисления ENUM_ACCOUNT_TRADE_MODE.

© 2000-2025, MetaQuotes Ltd.

TradeModeDescription

Получает режим торговли как строку.

```
string  TradeModeDescription() const
```

Возвращаемое значение

Режим торговли как строка.

© 2000-2025, MetaQuotes Ltd.

Leverage

Получает размер предоставленного кредитного плеча.

```
long  Leverage() const
```

Возвращаемое значение

Размер предоставленного кредитного плеча.

© 2000-2025, MetaQuotes Ltd.

StopoutMode

Получает режим задания минимального уровня залога.

```
ENUM_ACCOUNT_STOPOUT_MODE  StopoutMode() const
```

Возвращаемое значение

Режим задания минимального уровня залога из перечисления ENUM_ACCOUNT_STOPOUT_MODE.

© 2000-2025, MetaQuotes Ltd.

StopoutModeDescription

Получает режим задания минимального уровня залога как строку.

```
string  StopoutModeDescription() const
```

Возвращаемое значение

Режим задания минимального уровня залога как строка.

© 2000-2025, MetaQuotes Ltd.

MarginMode

Получает режим расчета маржи.

```
ENUM_ACCOUNT_MARGIN_MODE  MarginMode() const
```

Возвращаемое значение

Режим расчета маржи из перечисления ENUM_ACCOUNT_MARGIN_MODE.

© 2000-2025, MetaQuotes Ltd.

MarginModeDescription

Получает режим расчета маржи как строку.

```
string  MarginModeDescription() const
```

Возвращаемое значение

Режим расчета маржи в виде строки.

© 2000-2025, MetaQuotes Ltd.

TradeAllowed

Получает флаг разрешения торговли.

```
bool  TradeAllowed() const
```

Возвращаемое значение

Флаг разрешения торговли.

© 2000-2025, MetaQuotes Ltd.

TradeExpert

Получает флаг разрешения автоматизированной торговли.

```
bool  TradeExpert() const
```

Возвращаемое значение

Флаг разрешения автоматизированной торговли.

© 2000-2025, MetaQuotes Ltd.

LimitOrders

Получает максимально допустимое количество действующих отложенных ордеров.

```
int  LimitOrders() const
```

Возвращаемое значение

Максимально допустимое количество действующих отложенных ордеров.

Примечание

0 - ограничений нет.

© 2000-2025, MetaQuotes Ltd.

Balance

Получает баланс счета.

```
double  Balance() const
```

Возвращаемое значение

Баланс счета в валюте депозита.

© 2000-2025, MetaQuotes Ltd.

Credit

Получает размер предоставленного кредита.

```
double  Credit() const
```

Возвращаемое значение

Размер предоставленного кредита в валюте депозита.

© 2000-2025, MetaQuotes Ltd.

Profit

Получает значение текущей прибыли по счету.

```
double  Profit() const
```

Возвращаемое значение

Значение текущей прибыли по счету в валюте депозита.

© 2000-2025, MetaQuotes Ltd.

Equity

Получает значение собственных средств на счете.

```
double  Equity() const
```

Возвращаемое значение

Значение собственных средств на счете в валюте депозита.

© 2000-2025, MetaQuotes Ltd.

Margin

Получает размер зарезервированных залоговых средств.

```
double  Margin() const
```

Возвращаемое значение

Размер зарезервированных залоговых средств в валюте депозита.

© 2000-2025, MetaQuotes Ltd.

FreeMargin

Получает размер свободных средств.

```
double  FreeMargin() const
```

Возвращаемое значение

Размер свободных средств в валюте депозита.

© 2000-2025, MetaQuotes Ltd.

MarginLevel

Получает уровень залоговых средств.

```
double  MarginLevel() const
```

Возвращаемое значение

Уровень залоговых средств.

© 2000-2025, MetaQuotes Ltd.

MarginCall

Получает уровень залоговых средств пополнения счета.

```
double  MarginCall() const
```

Возвращаемое значение

Уровень залоговых средств пополнения счета.

© 2000-2025, MetaQuotes Ltd.

MarginStopOut

Получает уровень залоговых средств принудительного закрытия.

```
double  MarginStopOut() const
```

Возвращаемое значение

Уровень залоговых средств принудительного закрытия.

© 2000-2025, MetaQuotes Ltd.

Name

Получает имя клиента.

```
string  Name() const
```

Возвращаемое значение

Имя клиента.

© 2000-2025, MetaQuotes Ltd.

Server

Получает наименование торгового сервера.

```
string  Server() const
```

Возвращаемое значение

Наименование торгового сервера.

© 2000-2025, MetaQuotes Ltd.

Currency

Получает наименование валюты депозита.

```
string  Currency() const
```

Возвращаемое значение

Наименование валюты депозита.

© 2000-2025, MetaQuotes Ltd.

Company

Получает наименование компании, обслуживающей счет.

```
string  Company() const
```

Возвращаемое значение

Наименование компании, обслуживающей счет.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Получает значение указанного целочисленного свойства счета.

```
long  InfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  prop_id     // идентификатор свойства
   ) const
```

Параметры

prop_id

[in] Идентификатор свойства. Значение может быть одним из значений ENUM_ACCOUNT_INFO_INTEGER.

Возвращаемое значение

Значение типа long.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Получает значение указанного double-свойства счета.

```
double  InfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  prop_id     // идентификатор свойства
   ) const
```

Параметры

prop_id

[in] Идентификатор свойства. Значение может быть одним из значений ENUM_ACCOUNT_INFO_DOUBLE.

Возвращаемое значение

Значение типа double.

© 2000-2025, MetaQuotes Ltd.

InfoString

Получает значение указанного текстового свойства счета.

```
string  InfoString(
   ENUM_ACCOUNT_INFO_STRING  prop_id     // идентификатор свойства
   ) const
```

Параметры

prop_id

[in] Идентификатор свойства. Значение может быть одним из значений ENUM_ACCOUNT_INFO_STRING.

Возвращаемое значение

Значение типа string.

© 2000-2025, MetaQuotes Ltd.

OrderProfitCheck

Вычисляет размер прибыли для текущего счета на основании переданных параметров. Предназначена для предварительной оценки результата торговой операции. Значение возвращается в валюте счета.

```
double  OrderProfitCheck(
   const string        symbol,              // символ
   ENUM_ORDER_TYPE     trade_operation,     // тип ордера (ORDER_TYPE_BUY или ORDER_TY
   double              volume,              // объем
   double              price_open,          // цена открытия позиции
   double              price_close          // цена закрытия позиции
   ) const
```

Параметры

symbol

[in] Символ, по которому предполагается совершить торговую операцию.

trade_operation

[in] Тип торговой операции из перечисления ENUM_ORDER_TYPE.

volume

[in] Объем торговой операции.

price_open

[in] Цена открытия.

price_close

[in] Цена закрытия.

Возвращаемое значение

Возвращает размер прибыли в случае успеха, или EMPTY_VALUE в случае ошибки.

© 2000-2025, MetaQuotes Ltd.

MarginCheck

Получает размер маржи, необходимой для торговой операции.

```
double  MarginCheck(
   const string         symbol,              // символ
   ENUM_ORDER_TYPE      trade_operation,     // тип ордера (ORDER_TYPE_BUY или ORDER_TY
   double               volume,              // объем
   double               price                // цена
   ) const
```

Параметры

symbol

[in] Символ, по которому предполагается совершить торговую операцию.

trade_operation

[in] Тип торговой операции из перечисления ENUM_ORDER_TYPE.

volume

[in] Объем торговой операции.

price

[in] Цена торговой операции.

Возвращаемое значение

Размер маржи, необходимой для торговой операции.

© 2000-2025, MetaQuotes Ltd.

FreeMarginCheck

Получает размер свободной маржи, которая останется после торговой операции.

```
double  FreeMarginCheck(
   const string         symbol,              // символ
   ENUM_ORDER_TYPE      trade_operation,     // тип ордера (ORDER_TYPE_BUY или ORDER_TY
   double               volume,              // объем
   double               price                // цена
   ) const
```

Параметры

symbol

[in] Символ, по которому предполагается совершить торговую операцию.

trade_operation

[in] Тип торговой операции из перечисления ENUM_ORDER_TYPE.

volume

[in] Объем торговой операции.

price

[in] Цена торговой операции.

Возвращаемое значение

Размер свободной маржи, которая останется после торговой операции.

© 2000-2025, MetaQuotes Ltd.

MaxLotCheck

Получает максимально возможный объем торговой операции.

```
double  MaxLotCheck(
   const string         symbol,              // символ
   ENUM_ORDER_TYPE      trade_operation,     // тип ордера (ORDER_TYPE_BUY или ORDER_TY
   double               price                // цена
   double               percent=100          // доля свободной маржи (по умолчанию 100%
   ) const
```

Параметры

symbol

[in] Символ, по которому предполагается совершить торговую операцию.

trade_operation

[in] Тип торговой операции из перечисления ENUM_ORDER_TYPE.

price

[in] Цена торговой операции.

percent=100

[in] Доля свободной маржи (в процентах), которую предполагается использовать для осуществления торговой операции.

Возвращаемое значение

Максимально возможный объем торговой операции.

© 2000-2025, MetaQuotes Ltd.

Класс CSymbolInfo

Класс CSymbolInfo является классом для упрощенного доступа к свойствам символа.

Описание

Класс CSymbolInfo обеспечивает доступ к свойствам символа.

Декларация

```
class CSymbolInfo : public CObject
```

Заголовок

```
#include <Trade\SymbolInfo.mqh>
```

Иерархия наследования

CObject

CSymbolInfo

Методы класса по группам

Управление	
Refresh	Обновляет данные по символу
RefreshRates	Обновляет котировки по символу
Свойства	
Name	Получает/устанавливает наименование финансового инструмента
Select	Получает/устанавливает флаг символа "Символ в Обзоре Рынка"
IsSynchronized	Проверяет синхронизацию символа с сервером
Объемы	
Volume	Получает объем последней сделки
VolumeHigh	Получает максимальный объем за день
VolumeLow	Получает минимальный объем за день
Разное	
Time	Получает время последней котировки
Spread	Получает размер спреда в пунктах
SpreadFloat	Получает признак плавающего спреда
TicksBookDepth	Получает глубину хранения тиков

© 2000-2025, MetaQuotes Ltd.

Управление	
Уровни	
StopsLevel	Получает минимальный отступ для ордеров в пунктах
FreezeLevel	Получает дистанцию заморозки торговых операций в пунктах
Цены спроса	
Bid	Получает текущую цену Bid
BidHigh	Получает максимальную цену Bid за день
BidLow	Получает минимальную цену Bid за день
Цены предложения	
Ask	Получает текущую цену Ask
AskHigh	Получает максимальную цену Ask за день
AskLow	Получает минимальную цену Ask за день
Цены	
Last	Получает текущую цену Last
LastHigh	Получает максимальную цену Last за день
LastLow	Получает минимальную цену Last за день
Режимы торговли	
TradeCalcMode	Получает способ вычисления стоимости контрактов
TradeCalcModeDescription	Получает способ вычисления стоимости контрактов как строку
TradeMode	Получает тип исполнения ордеров
TradeModeDescription	Получает тип исполнения ордеров как строку
TradeExecution	Получает режим заключения сделок
TradeExecutionDescription	Получает режим заключения сделок как строку
Свопы	
SwapMode	Получает модель расчета свопа
SwapModeDescription	Получает модель расчета свопа как строку
SwapRollover3days	Получает день недели начисления тройного свопа
SwapRollover3daysDescription	Получает день недели начисления тройного свопа как строку
Залоги и флаги	
MarginInitial	Получает значение начальной маржи
MarginMaintenance	Получает значение поддерживающей маржи

© 2000-2025, MetaQuotes Ltd.

Управление	
MarginLong	Получает коэффициент взимания маржи по длинным позициям
MarginShort	Получает коэффициент взимания маржи по коротким позициям
MarginLimit	Получает коэффициент взимания маржи по Limit ордерам
MarginStop	Получает коэффициент взимания маржи по Stop ордерам
MarginStopLimit	Получает коэффициент взимания маржи по Stop Limit ордерам
TradeTimeFlags	Получает флаги разрешённых режимов истечения ордера
TradeFillFlags	Получает флаги разрешённых режимов заливки ордера
Квантование	
Digits	Получает количество знаков после десятичной точки
Point	Получает значение одного пункта
TickValue	Получает стоимость минимального изменения цены
TickValueProfit	Получает рассчитанную стоимость тика для прибыльной позиции
TickValueLoss	Получает рассчитанную стоимость тика для убыточной позиции
TickSize	Получает минимальное изменение цены
Размеры контрактов	
ContractSize	Получает размер торгового контракта
LotsMin	Получает минимальный объем для заключения сделки
LotsMax	Получает максимальный объем для заключения сделки
LotsStep	Получает минимальный шаг изменения объема для заключения сделки
LotsLimit	Получает максимально допустимый совокупный объем открытой позиции и отложенных ордеров на символе
Размеры свопов	
SwapLong	Получает значение свопа длинной позиции
SwapShort	Получает значение свопа короткой позиции
Текстовые свойства	
CurrencyBase	Получает наименование базовой валюты символа
CurrencyProfit	Получает наименование валюты прибыли
CurrencyMargin	Получает наименование валюты залога
Bank	Получает наименование источника текущей котировки

© 2000-2025, MetaQuotes Ltd.

Управление	
Description	Получает строковое описание символа
Path	Получает путь в дереве символов
Свойства символов	
SessionDeals	Получает количество сделок в текущей сессии
SessionBuyOrders	Получает общее число ордеров на покупку в текущий момент
SessionSellOrders	Получает общее число ордеров на продажу в текущий момент
SessionTurnover	Получает суммарный оборот в текущую сессию
SessionInterest	Получает суммарный объем открытых позиций
SessionBuyOrdersVolume	Получает общий объем ордеров на покупку в текущий момент
SessionSellOrdersVolume	Получает общий объем ордеров на продажу в текущий момент
SessionOpen	Получает цену открытия текущей сессии
SessionClose	Получает цену закрытия текущей сессии
SessionAW	Получает средневзвешенную цену текущей сессии
SessionPriceSettlement	Получает цену поставки на текущую сессию
SessionPriceLimitMin	Получает минимально допустимое значение цены за текущую сессию
SessionPriceLimitMax	Получает максимально допустимое значение цены за текущую сессию
Доступ к функциям API MQL5	
InfoInteger	Получает значение указанного целочисленного свойства
InfoDouble	Получает значение указанного double-свойства
InfoString	Получить значение указанного текстового свойства
Сервисные функции	
NormalizePrice	Нормализует цену с учетом свойств символа

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет данные по символу.

```
void  Refresh()
```

Возвращаемое значение

Нет.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

RefreshRates

Обновляет котировки по символу.

```
bool  RefreshRates()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось обновить котировки.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Name

Получает наименование финансового инструмента.

```
string  Name() const
```

Возвращаемое значение

Наименование финансового инструмента.

Name

Устанавливает наименование финансового инструмента для дальнейшей работы с ним.

```
bool  Name(string name)
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Select

Получает флаг символа "Символ в обзоре рынка".

```
bool  Select() const
```

Возвращаемое значение

Флаг символа "Символ в обзоре рынка".

Select

Устанавливает флаг символа "Символ в обзоре рынка".

```
bool  Select()
```

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить флаг.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

IsSynchronized

Проверяет синхронизацию символа с сервером.

```
bool  IsSynchronized() const
```

Возвращаемое значение

true - если символ синхронизирован с сервером, false - если нет.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Volume

Получает объем последней сделки.

```
long  Volume() const
```

Возвращаемое значение

Объем последней сделки.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

VolumeHigh

Получает максимальный объем за день.

```
long  VolumeHigh() const
```

Возвращаемое значение

Максимальный объем за день.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

VolumeLow

Получает минимальный объем за день.

```
long  VolumeLow() const
```

Возвращаемое значение

Минимальный объем за день.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Time

Получает время последней котировки.

```
datetime  Time() const
```

Возвращаемое значение

Время последней котировки.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Spread

Получает размер спреда в пунктах.

```
int  Spread() const
```

Возвращаемое значение

Размер спреда в пунктах.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SpreadFloat

Получает признак плавающего спреда.

```
bool  SpreadFloat() const
```

Возвращаемое значение

Признак плавающего спреда.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TicksBookDepth

Получает глубину хранения тиков.

```
int  TicksBookDepth() const
```

Возвращаемое значение

Глубина хранения тиков.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

StopsLevel

Получает минимальный отступ для ордеров в пунктах.

```
int  StopsLevel() const
```

Возвращаемое значение

Минимальный отступ для ордеров в пунктах.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

FreezeLevel

Получает дистанцию заморозки торговых операций в пунктах.

```
int  FreezeLevel() const
```

Возвращаемое значение

Дистанция заморозки торговых операций в пунктах.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Bid

Получает текущую цену Bid.

```
double  Bid() const
```

Возвращаемое значение

Текущая цена Bid.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

BidHigh

Получает максимальную цену Bid за день.

```
double  BidHigh() const
```

Возвращаемое значение

Максимальная цена Bid за день.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

BidLow

Получает минимальную цену Bid за день.

```
double  BidLow() const
```

Возвращаемое значение

Минимальная цена Bid за день.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Ask

Получает текущую цену Ask.

```
double  Ask() const
```

Возвращаемое значение

Текущая цена Ask.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

AskHigh

Получает максимальную цену Ask за день.

```
double  AskHigh() const
```

Возвращаемое значение

Максимальная цена Ask за день.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

AskLow

Получает минимальную цену Ask за день.

```
double  AskLow() const
```

Возвращаемое значение

Минимальная цена Ask за день.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Last

Получает текущую цену Last.

```
double  Last() const
```

Возвращаемое значение

Текущая цена Last.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

LastHigh

Получает максимальную цену Last за день.

```
double  LastHigh() const
```

Возвращаемое значение

Максимальная цена Last за день.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

LastLow

Получает минимальную цену Last за день.

```
double  LastLow() const
```

Возвращаемое значение

Минимальная цена Last за день.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TradeCalcMode

Получает способ вычисления стоимости контрактов.

```
ENUM_SYMBOL_CALC_MODE  TradeCalcMode() const
```

Возвращаемое значение

Способ вычисления стоимости контрактов из перечисления ENUM_SYMBOL_CALC_MODE.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TradeCalcModeDescription

Получает способ вычисления стоимости контрактов как строку.

```
string  TradeCalcModeDescription() const
```

Возвращаемое значение

Способ вычисления стоимости контрактов как строка.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TradeMode

Получает тип исполнения ордеров.

```
ENUM_SYMBOL_TRADE_MODE  TradeMode() const
```

Возвращаемое значение

Тип исполнения ордеров из перечисления ENUM_SYMBOL_TRADE_MODE.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TradeModeDescription

Получает тип исполнения ордеров как строку.

```
string  TradeModeDescription() const
```

Возвращаемое значение

Тип исполнения ордеров как строка.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TradeExecution

Получает режим заключения сделок.

```
ENUM_SYMBOL_TRADE_EXECUTION  TradeExecution() const
```

Возвращаемое значение

Режим заключения сделок из перечисления ENUM_SYMBOL_TRADE_EXECUTION.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TradeExecutionDescription

Получает режим заключения сделок как строку.

```
string  TradeExecutionDescription() const
```

Возвращаемое значение

Режим заключения сделок как строка.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SwapMode

Получает модель расчета свопа.

```
ENUM_SYMBOL_SWAP_MODE  SwapMode() const
```

Возвращаемое значение

Модель расчета свопа из перечисления ENUM_SYMBOL_SWAP_MODE.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SwapModeDescription

Получает модель расчета свопа как строку.

```
string  SwapModeDescription() const
```

Возвращаемое значение

Модель расчета свопа как строка.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SwapRollover3days

Получает день недели начисления тройного свопа.

```
ENUM_DAY_OF_WEEK  SwapRollover3days() const
```

Возвращаемое значение

День недели начисления тройного свопа из перечисления ENUM_DAY_OF_WEEK.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SwapRollover3daysDescription

Получает день недели начисления тройного свопа как строку.

```
string  SwapRollover3daysDescription() const
```

Возвращаемое значение

День недели начисления тройного свопа как строка.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

MarginInitial

Получает значение начальной маржи.

```
double  MarginInitial()
```

Возвращаемое значение

Значение начальной маржи.

Примечание

Указывает размер маржи в маржинальной валюте инструмента, взимаемую с одного лота. Используется при проверке средств клиента при входе в рынок.

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

MarginMaintenance

Получает значение поддерживающей маржи.

```
double  MarginMaintenance()
```

Возвращаемое значение

Значение поддерживающей маржи.

Примечание

Указывает размер маржи в маржинальной валюте инструмента, взимаемой с одного лота. Используется для проверки средств клиента, при изменении состояния счёта. Если поддерживающая маржа равна 0, то используется начальная маржа.

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

MarginLong

Получает коэффициент взимания маржи по длинным позициям.

```
double  MarginLong() const
```

Возвращаемое значение

Коэффициент взимания маржи по длинным позициям.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

MarginShort

Получает коэффициент взимания маржи по коротким позициям.

```
double  MarginShort() const
```

Возвращаемое значение

Коэффициент взимания маржи по коротким позициям.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

MarginLimit

Получает коэффициент взимания маржи по Limit ордерам.

```
double  MarginLimit() const
```

Возвращаемое значение

Коэффициент взимания маржи по Limit ордерам.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

MarginStop

Получает коэффициент взимания маржи по Stop ордерам.

```
double  MarginStop() const
```

Возвращаемое значение

Коэффициент взимания маржи по Stop ордерам.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

MarginStopLimit

Получает коэффициент взимания маржи по Stop Limit ордерам.

```
double  MarginStopLimit() const
```

Возвращаемое значение

Коэффициент взимания маржи по Stop Limit ордерам.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TradeTimeFlags

Получает флаги разрешенных режимов истечения ордера.

```
int  TradeTimeFlags() const
```

Возвращаемое значение

Флаги разрешенных режимов истечения ордера.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TradeFillFlags

Получает флаги разрешенных режимов заливки ордера.

```
int  TradeFillFlags() const
```

Возвращаемое значение

Флаги разрешенных режимов заливки ордера.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Digits

Получает количество знаков после десятичной точки.

```
int  Digits() const
```

Возвращаемое значение

Количество знаков после десятичной точки.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Point

Получает значение одного пункта.

```
double  Point() const
```

Возвращаемое значение

Значение одного пункта.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TickValue

Получает стоимость минимального изменения цены.

```
double  TickValue() const
```

Возвращаемое значение

Стоимость минимального изменения цены.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TickValueProfit

Получает рассчитанную стоимость тика для прибыльной позиции.

```
double  TickValueProfit() const
```

Возвращаемое значение

Рассчитанная стоимость тика для прибыльной позиции.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TickValueLoss

Получает рассчитанную стоимость тика для убыточной позиции.

```
double  TickValueLoss() const
```

Возвращаемое значение

Рассчитанная стоимость тика для убыточной позиции.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

TickSize

Получает минимальное изменение цены.

```
double  TickSize() const
```

Возвращаемое значение

Минимальное изменение цены.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

ContractSize

Получает размер торгового контракта.

```
double  ContractSize() const
```

Возвращаемое значение

Размер торгового контракта.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

LotsMin

Получает минимальный объем для заключения сделки.

```
double  LotsMin() const
```

Возвращаемое значение

Минимальный объем для заключения сделки.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

LotsMax

Получает максимальный объем для заключения сделки.

```
double  LotsMax() const
```

Возвращаемое значение

Максимальный объем для заключения сделки.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

LotsStep

Получает минимальный шаг изменения объема для заключения сделки.

```
double  LotsStep() const
```

Возвращаемое значение

Минимальный шаг изменения объема для заключения сделки.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

LotsLimit

Получает максимально допустимый совокупный объем открытой позиции и отложенных ордеров (вне зависимости от направления) на одном символе.

```
double  LotsLimit() const
```

Возвращаемое значение

Максимально допустимый совокупный объем открытой позиции и отложенных ордеров (вне зависимости от направления) на одном символе.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SwapLong

Получает значение свопа длинной позиции.

```
double  SwapLong() const
```

Возвращаемое значение

Значение свопа длинной позиции.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SwapShort

Получает значение свопа короткой позиции.

```
double  SwapShort() const
```

Возвращаемое значение

Значение свопа короткой позиции.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

CurrencyBase

Получает наименование базовой валюты символа.

```
string  CurrencyBase() const
```

Возвращаемое значение

Наименование базовой валюты символа.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

CurrencyProfit

Получает наименование валюты прибыли.

```
string  CurrencyProfit() const
```

Возвращаемое значение

Наименование валюты прибыли.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

CurrencyMargin

Получает наименование валюты залога.

```
string  CurrencyMargin() const
```

Возвращаемое значение

Наименование валюты залога.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Bank

Получает наименование источника текущей котировки.

```
string  Bank() const
```

Возвращаемое значение

Наименование источника текущей котировки.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Description

Получает строковое описание символа.

```
string  Description() const
```

Возвращаемое значение

Строковое описание символа.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Path

Получает путь в дереве символов.

```
string  Path() const
```

Возвращаемое значение

Путь в дереве символов.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionDeals

Получает количество сделок в текущей сессии.

```
long  SessionDeals() const
```

Возвращаемое значение

Количество сделок в текущей сессии.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionBuyOrders

Получает общее число ордеров на покупку в текущий момент.

```
long  SessionBuyOrders() const
```

Возвращаемое значение

Общее число ордеров на покупку в текущий момент.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionSellOrders

Получает общее число ордеров на продажу в текущий момент.

```
long  SessionSellOrders() const
```

Возвращаемое значение

Общее число ордеров на продажу в текущий момент.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionTurnover

Получает суммарный оборот в текущую сессию.

```
double  SessionTurnover() const
```

Возвращаемое значение

Суммарный оборот в текущую сессию.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionInterest

Получает суммарный объем открытых позиций.

```
double  SessionInterest() const
```

Возвращаемое значение

Суммарный объем открытых позиций.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionBuyOrdersVolume

Получает общий объем ордеров на покупку в текущий момент.

```
double  SessionBuyOrdersVolume() const
```

Возвращаемое значение

Общий объем ордеров на покупку в текущий момент.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionSellOrdersVolume

Получает общий объем ордеров на продажу в текущий момент.

```
double  SessionSellOrdersVolume() const
```

Возвращаемое значение

Общий объем ордеров на продажу в текущий момент.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionOpen

Получает цену открытия текущей сессии.

```
double  SessionOpen() const
```

Возвращаемое значение

Цена открытия текущей сессии.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionClose

Получает цену закрытия текущей сессии.

```
double  SessionClose() const
```

Возвращаемое значение

Цена закрытия текущей сессии.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionAW

Получает значение средневзвешенной цены текущей сессии.

```
double  SessionAW() const
```

Возвращаемое значение

Значение средневзвешенной цены текущей сессии.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionPriceSettlement

Получает значение цены поставки на текущую сессию.

```
double  SessionPriceSettlement() const
```

Возвращаемое значение

Значение цены поставки на текущую сессию.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionPriceLimitMin

Получает минимально допустимое значение цены за текущую сессию.

```
double  SessionPriceLimitMin() const
```

Возвращаемое значение

Минимально допустимое значение цены за текущую сессию.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

SessionPriceLimitMax

Получает максимально допустимое значение цены на сессию.

```
double  SessionPriceLimitMax() const
```

Возвращаемое значение

Максимально допустимое значение цены на сессию.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Получает значение указанного целочисленного свойства.

```
bool  InfoInteger(
   ENUM_SYMBOL_INFO_INTEGER  prop_id,      // идентификатор свойства
   long&                     var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор целочисленного свойства из перечисления ENUM_SYMBOL_INFO_INTEGER.

var

[out] Ссылка на переменную типа long для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Получает значение указанного double-свойства.

```
bool  InfoDouble(
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,      // идентификатор свойства
   double&                  var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор double-свойства из перечисления ENUM_SYMBOL_INFO_DOUBLE.

var

[out] Ссылка на переменную типа double для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

InfoString

Получает значение указанного текстового свойства.

```
bool  InfoString(
   ENUM_SYMBOL_INFO_STRING  prop_id,      // идентификатор свойства
   string&                  var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор текстового свойства.

var

[out] Ссылка на переменную типа string для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

NormalizePrice

Нормализует цену с учетом свойств символа.

```
double  NormalizePrice(
   double  price   // цена
   ) const
```

Параметры

price

[in] Цена.

Возвращаемое значение

Нормализованная цена.

Примечание

Символ должен быть предварительно выбран методом Name.

© 2000-2025, MetaQuotes Ltd.

Класс COrderInfo

Класс COrderInfo является классом для упрощенного доступа к свойствам отложенного ордера.

Описание

Класс COrderInfo обеспечивает доступ к свойствам отложенного ордера.

Декларация

```
class COrderInfo : public CObject
```

Заголовок

```
#include <Trade\OrderInfo.mqh>
```

Иерархия наследования

CObject

COrderInfo

Методы класса по группам

Доступ к целочисленным свойствам	
Ticket	Получает тикет ордера, предварительно выбранного для доступа
TimeSetup	Получает время установки ордера
TimeSetupMsc	Получает время установки ордера в миллисекундах с 01.01.1970
OrderType	Получает тип ордера
TypeDescription	Получает тип ордера как строку
State	Получает статус ордера
StateDescription	Получает статус ордера как строку
TimeExpiration	Получает время истечения ордера
TimeDone	Получает время исполнения или снятия ордера
TimeDoneMsc	Получает время исполнения или снятия ордера в миллисекундах с 01.01.1970
TypeFilling	Получает тип исполнения ордера по остатку
TypeFillingDescription	Получает тип исполнения ордера по остатку как строку
TypeTime	Получает тип ордера по времени истечения
TypeTimeDescription	Получает тип ордера по времени истечения как строку

© 2000-2025, MetaQuotes Ltd.

Доступ к целочисленным свойствам	
Magic	Получает идентификатор эксперта, выставившего ордер
PositionId	Получает идентификатор позиции
Доступ к double-свойствам	
VolumeInitial	Получает первоначальный объем ордера
VolumeCurrent	Получает невыполненный объем ордера
PriceOpen	Получает цену ордера
StopLoss	Получает стоп лосс ордера
TakeProfit	Получает тейк профит ордера
PriceCurrent	Получает текущую цену по символу ордера
PriceStopLimit	Получает цену постановки лимит ордера
Доступ к текстовым свойствам	
Symbol	Получает наименование символа ордера
Comment	Получает комментарий ордера
Доступ к функциям API MQL5	
InfoInteger	Получает значение указанного целочисленного свойства
InfoDouble	Получает значение указанного double-свойства
InfoString	Получает значение указанного текстового свойства
Состояние	
StoreState	Сохраняет параметры ордера
CheckState	Сравнивает текущие параметры с сохраненными
Выбор	
Select	Выбирает ордер для дальнейшего доступа по указанному тикету
SelectByIndex	Выбирает ордер для дальнейшего доступа по указанному индексу

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Ticket

Получает тикет ордера.

```
ulong  Ticket() const
```

Возвращаемое значение

Тикет ордера в случае успеха, или ULONG_MAX в случае ошибки.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeSetup

Получает время установки ордера.

```
datetime  TimeSetup() const
```

Возвращаемое значение

Время установки ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeSetupMsc

Получает время установки ордера на исполнение в миллисекундах с 01.01.1970.

```
ulong  TimeSetupMsc() const
```

Возвращаемое значение

Время установки ордера на исполнение в миллисекундах с 01.01.1970.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

OrderType

Получает тип ордера.

```
ENUM_ORDER_TYPE  OrderType()
```

Возвращаемое значение

Тип ордера из перечисления ENUM_ORDER_TYPE.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeDescription

Получает тип ордера как строку.

```
string  TypeDescription() const
```

Возвращаемое значение

Тип ордера как строка.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

State

Получает статус ордера.

```
ENUM_ORDER_STATE  State() const
```

Возвращаемое значение

Статус ордера из перечисления ENUM_ORDER_STATE.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

StateDescription

Получает статус ордера как строку.

```
string  StateDescription() const
```

Возвращаемое значение

Статус ордера как строка.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeExpiration

Получает время истечения ордера.

```
datetime  TimeExpiration() const
```

Возвращаемое значение

Заданное при установке время истечения ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeDone

Получает время исполнения или снятия ордера.

```
datetime  TimeDone() const
```

Возвращаемое значение

Время исполнения или снятия ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeDoneMsc

Получает время исполнения или снятия ордера в миллисекундах с 01.01.1970.

```
ulong  TimeDoneMsc() const
```

Возвращаемое значение

Время исполнения или снятия ордера в миллисекундах с 01.01.1970.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeFilling

Получает тип исполнения ордера по остатку.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Возвращаемое значение

Тип исполнения ордера по остатку из перечисления ENUM_ORDER_TYPE_FILLING.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeFillingDescription

Получает тип исполнения ордера по остатку как строку.

```
string  TypeFillingDescription() const
```

Возвращаемое значение

Тип исполнения ордера по остатку как строка.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeTime

Получает тип ордера по времени истечения.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Возвращаемое значение

Тип ордера по времени истечения из перечисления ENUM_ORDER_TYPE_TIME.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeTimeDescription

Получает тип ордера по времени истечения как строку.

```
string  TypeTimeDescription() const
```

Возвращаемое значение

Тип ордера по времени истечения как строка.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Magic

Получает идентификатор эксперта, выставившего ордер.

```
long  Magic() const
```

Возвращаемое значение

Идентификатор эксперта, выставившего ордер.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PositionId

Получает идентификатор позиции.

```
long  PositionId() const
```

Возвращаемое значение

Идентификатор позиции, в которой участвует ордер.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

VolumeInitial

Получает первоначальный объем ордера.

```
double  VolumeInitial() const
```

Возвращаемое значение

Первоначальный объем ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

VolumeCurrent

Получает невыполненный объем ордера.

```
double  VolumeCurrent() const
```

Возвращаемое значение

Невыполненный объем ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PriceOpen

Получает цену ордера.

```
double  PriceOpen() const
```

Возвращаемое значение

Цена установки ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

StopLoss

Получает цену Stop Loss ордера.

```
double  StopLoss() const
```

Возвращаемое значение

Цена Stop Loss ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TakeProfit

Получает цену Take Profit ордера.

```
double  TakeProfit() const
```

Возвращаемое значение

Цена Take Profit ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PriceCurrent

Получает текущую цену по символу ордера.

```
double  PriceCurrent() const
```

Возвращаемое значение

Текущая цена по символу ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PriceStopLimit

Получает цену постановки отложенного ордера.

```
double  PriceStopLimit() const
```

Возвращаемое значение

Цена постановки отложенного ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Symbol

Получает наименование символа ордера.

```
string  Symbol() const
```

Возвращаемое значение

Наименование символа ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Comment

Получает комментарий ордера.

```
string  Comment() const
```

Возвращаемое значение

Комментарий ордера.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Получает значение указанного целочисленного свойства.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // идентификатор свойства
   long&                        var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор целочисленного свойства из перечисления ENUM_ORDER_PROPERTY_INTEGER.

var

[out] Ссылка на переменную типа long для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Получает значение указанного double-свойства.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // идентификатор свойства
   double&                     var          // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор double-свойства из перечисления ENUM_ORDER_PROPERTY_DOUBLE.

var

[out] Ссылка на переменную типа double для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoString

Получает значение указанного текстового свойства.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // идентификатор свойства
   string&                     var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор текстового свойства из перечисления ENUM_ORDER_PROPERTY_STRING.

var

[out] Ссылка на переменную типа string для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Ордер должен быть предварительно выбран для доступа методом Select (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

StoreState

Сохраняет параметры ордера.

```
void  StoreState()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

CheckState

Сравнивает текущие параметры с сохраненными.

```
bool  CheckState()
```

Возвращаемое значение

true - если параметры ордера изменились за время, прошедшее после последнего вызова метода StoreState(), иначе false.

© 2000-2025, MetaQuotes Ltd.

Select

Выбирает ордер для дальнейшего доступа по указанному тикету.

```
bool  Select(
   ulong  ticket   // тикет ордера
   )
```

Возвращаемое значение

Возвращает true в случае успеха, или false, если ордер не выбран.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

Выбирает ордер по индексу для дальнейшей работы.

```
bool  SelectByIndex(
   int  index      // индекс ордера
   )
```

Параметры

index

[in] Индекс ордера.

Возвращаемое значение

Возвращает true в случае успеха, или false, если ордер не выбран.

© 2000-2025, MetaQuotes Ltd.

Класс CHistoryOrderInfo

Класс CHistoryOrderInfo является классом для упрощенного доступа к свойствам ордера истории.

Описание

Класс CHistoryOrderInfo обеспечивает доступ к свойствам ордера истории.

Декларация

```
class CHistoryOrderInfo : public CObject
```

Заголовок

```
#include <Trade\HistoryOrderInfo.mqh>
```

Иерархия наследования

CObject

CHistoryOrderInfo

Методы класса по группам

Доступ к целочисленным свойствам	
TimeSetup	Получает время установки ордера
TimeSetupMsc	Получает время установки ордера в миллисекундах с 01.01.1970
OrderType	Получает тип ордера
TypeDescription	Получает тип ордера как строку
State	Получает статус ордера
StateDescription	Получает статус ордера как строку
TimeExpiration	Получает время истечения ордера
TimeDone	Получает время исполнения или снятия ордера
TimeDoneMsc	Получает время исполнения или снятия ордера в миллисекундах с 01.01.1970
TypeFilling	Получает тип исполнения ордера по остатку
TypeFillingDescription	Получает тип исполнения ордера по остатку как строку
TypeTime	Получает тип ордера по времени истечения
TypeTimeDescription	Получает тип ордера по времени истечения как строку
Magic	Получает идентификатор эксперта, выставившего ордер
PositionId	Получает идентификатор позиции

© 2000-2025, MetaQuotes Ltd.

Доступ к целочисленным свойствам	
Доступ к double-свойствам	
VolumeInitial	Получает первоначальный объем ордера
VolumeCurrent	Получает невыполненный объем ордера
PriceOpen	Получает цену ордера
StopLoss	Получает стоп лосс ордера
TakeProfit	Получает тейк профит ордера
PriceCurrent	Получает текущую цену по символу ордера
PriceStopLimit	Получает цену постановки лимит ордера
Доступ к текстовым свойствам	
Symbol	Получает символ ордера
Comment	Получает комментарий ордера
Доступ к функциям API MQL5	
InfoInteger	Получает значение указанного целочисленного свойства
InfoDouble	Получает значение указанного double-свойства
InfoString	Получает значение указанного текстового свойства
Выбор	
Ticket	Получает тикет/выбирает ордер
SelectByIndex	Выбирает исторический ордер по индексу

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

TimeSetup

Получает время установки ордера.

```
datetime  TimeSetup() const
```

Возвращаемое значение

Время установки ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeSetupMsc

Получает время установки ордера на исполнение в миллисекундах с 01.01.1970.

```
ulong  TimeSetupMsc() const
```

Возвращаемое значение

Время установки ордера на исполнение в миллисекундах с 01.01.1970.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

OrderType

Получает тип ордера.

```
ENUM_ORDER_TYPE  OrderType() const
```

Возвращаемое значение

Тип ордера из перечисления ENUM_ORDER_TYPE.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeDescription

Получает тип ордера как строку.

```
string  TypeDescription() const
```

Возвращаемое значение

Тип ордера как строка.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

State

Получает статус ордера.

```
ENUM_ORDER_STATE  State() const
```

Возвращаемое значение

Статус ордера из перечисления ENUM_ORDER_STATE.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

StateDescription

Получает статус ордера как строку.

```
string  StateDescription() const
```

Возвращаемое значение

Статус ордера как строка.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeExpiration

Получает время истечения ордера.

```
datetime  TimeExpiration() const
```

Возвращаемое значение

Заданное при установке время истечения ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeDone

Получает время исполнения или снятия ордера.

```
datetime  TimeDone() const
```

Возвращаемое значение

Время исполнения или снятия ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeDoneMsc

Получает время исполнения или снятия ордера в миллисекундах с 01.01.1970.

```
ulong  TimeDoneMsc() const
```

Возвращаемое значение

Время исполнения или снятия ордера в миллисекундах с 01.01.1970.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeFilling

Получает тип исполнения ордера по остатку.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Возвращаемое значение

Тип исполнения ордера по остатку из перечисления ENUM_ORDER_TYPE_FILLING.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeFillingDescription

Получает тип исполнения ордера по остатку как строку.

```
string  TypeFillingDescription() const
```

Возвращаемое значение

Тип исполнения ордера по остатку как строка.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeTime

Получает тип ордера по времени истечения.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Возвращаемое значение

Тип ордера по времени истечения из перечисления ENUM_ORDER_TYPE_TIME.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeTimeDescription

Получает тип ордера по времени истечения как строку.

```
string  TypeTimeDescription() const
```

Возвращаемое значение

Тип ордера по времени истечения как строка.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Magic

Получает идентификатор эксперта, выставившего ордер.

```
long  Magic() const
```

Возвращаемое значение

Идентификатор эксперта, выставившего ордер.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PositionId

Получает идентификатор позиции.

```
long  PositionId() const
```

Возвращаемое значение

Идентификатор позиции, в которой участвовал ордер.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

VolumeInitial

Получает первоначальный объем ордера.

```
double  VolumeInitial() const
```

Возвращаемое значение

Первоначальный объем ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

VolumeCurrent

Получает невыполненный объем ордера.

```
double  VolumeCurrent() const
```

Возвращаемое значение

Невыполненный объем ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PriceOpen

Получает цену ордера.

```
double  PriceOpen() const
```

Возвращаемое значение

Цена установки ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

StopLoss

Получает цену Stop Loss ордера.

```
double  StopLoss() const
```

Возвращаемое значение

Цена Stop Loss ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TakeProfit

Получает цену Take Profit ордера.

```
double  TakeProfit() const
```

Возвращаемое значение

Цена Take Profit ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PriceCurrent

Получает текущую цену по символу ордера.

```
double  PriceCurrent() const
```

Возвращаемое значение

Текущая цена по символу ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PriceStopLimit

Получает цену постановки отложенного ордера.

```
double  PriceStopLimit() const
```

Возвращаемое значение

Цена постановки отложенного ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Symbol

Получает наименование символа ордера.

```
string  Symbol() const
```

Возвращаемое значение

Наименование символа ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Comment

Получает комментарий ордера.

```
string  Comment() const
```

Возвращаемое значение

Комментарий ордера.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Получает значение указанного целочисленного свойства.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // идентификатор свойства
   long&                        var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор целочисленного свойства из перечисления ENUM_ORDER_PROPERTY_INTEGER.

var

[out] Ссылка на переменную типа long для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Получает значение указанного double-свойства.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // идентификатор свойства
   double&                     var          // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор double-свойства из перечисления ENUM_ORDER_PROPERTY_DOUBLE.

var

[out] Ссылка на переменную типа double для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoString

Получает значение указанного текстового свойства.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // идентификатор свойства
   string&                     var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор текстового свойства из перечисления ENUM_ORDER_PROPERTY_STRING.

var

[out] Ссылка на переменную типа string для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Исторический ордер должен быть предварительно выбран для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Ticket (метод Get)

Получает тикет ордера.

```
ulong  Ticket() const
```

Возвращаемое значение

Тикет ордера.

Ticket (метод Set)

Выбирает ордер для дальнейшей работы.

```
void  Ticket(
   ulong  ticket      // тикет
   )
```

Параметры

ticket

[in] Тикет ордера.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

Выбирает исторический ордер по индексу для дальнейшей работы.

```
bool  SelectByIndex(
   int  index      // индекс ордера
   )
```

Параметры

index

[in] Индекс исторического ордера.

Возвращаемое значение

Возвращает true в случае успеха, или false, если ордер не выбран.

© 2000-2025, MetaQuotes Ltd.

Класс CPositionInfo

Класс CPositionInfo является классом для упрощенного доступа к свойствам открытой рыночной позиции.

Описание

Класс CPositionInfo обеспечивает доступ к свойствам открытой рыночной позиции.

Декларация

```
class CPositionInfo : public CObject
```

Заголовок

```
#include <Trade\PositionInfo.mqh>
```

Иерархия наследования

CObject

CPositionInfo

Методы класса по группам

Доступ к целочисленным свойствам	
Time	Получает время открытия позиции
TimeMsc	Получает время открытия позиции в миллисекундах с 01.01.1970
TimeUpdate	Получает время изменения позиции в секундах с 01.01.1970
TimeUpdateMsc	Получает время изменения позиции в миллисекундах с 01.01.1970
PositionType	Получает тип позиции
TypeDescription	Получает тип позиции как строку
Magic	Получает идентификатор эксперта, открывшего позицию
Identifier	Получает идентификатор позиции
Доступ к double-свойствам	
Volume	Получает объем позиции
PriceOpen	Получает цену открытия позиции
StopLoss	Получает цену стоп лосса позиции
TakeProfit	Получает цену тейк профита позиции
PriceCurrent	Получает текущую цену по символу позиции

© 2000-2025, MetaQuotes Ltd.

Доступ к целочисленным свойствам	
Commission	Получает размер комиссии по позиции
Swap	Получает размер свопа по позиции
Profit	Получает размер текущей прибыли по позиции
Доступ к текстовым свойствам	
Symbol	Получает наименование символа позиции
Comment	Получает комментарий позиции
Доступ к функциям API MQL5	
InfoInteger	Получает значение указанного целочисленного свойства
InfoDouble	Получает значение указанного double-свойства
InfoString	Получить значение указанного текстового свойства
Выбор	
Select	Выбирает позицию
SelectByIndex	Выбирает позицию по индексу
SelectByMagic	Выбирает позицию по имени инструмента и магическому номеру для дальнейшей работы
SelectByTicket	Выбирает позицию по тикету
Состояние	
StoreState	Сохраняет параметры позиции
CheckState	Сравнивает текущие параметры с сохраненными

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Time

Получает время открытия позиции.

```
datetime  Time() const
```

Возвращаемое значение

Время открытия позиции.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeMsc

Получает время открытия позиции в миллисекундах с 01.01.1970.

```
ulong  TimeMsc() const
```

Возвращаемое значение

Время открытия позиции в миллисекундах с 01.01.1970.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeUpdate

Получает время изменения позиции в секундах с 01.01.1970.

```
datetime  TimeUpdate() const
```

Возвращаемое значение

Время изменения позиции в секундах с 01.01.1970.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeUpdateMsc

Получает время изменения позиции в миллисекундах с 01.01.1970.

```
ulong  TimeUpdateMsc() const
```

Возвращаемое значение

Время изменения позиции в миллисекундах с 01.01.1970.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PositionType

Получает тип позиции.

```
ENUM_POSITION_TYPE  PositionType() const
```

Возвращаемое значение

Тип позиции из перечисления ENUM_POSITION_TYPE.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeDescription

Получает тип позиции как строку.

```
string  TypeDescription() const
```

Возвращаемое значение

Тип позиции как строка.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Magic

Получает идентификатор эксперта, открывшего позицию.

```
long  Magic() const
```

Возвращаемое значение

Идентификатор эксперта, открывшего позицию.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Identifier

Получает идентификатор позиции.

```
long  Identifier() const
```

Возвращаемое значение

Идентификатор позиции.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Volume

Получает объем позиции.

```
double  Volume() const
```

Возвращаемое значение

Объем позиции.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PriceOpen

Получает цену открытия позиции.

```
double  PriceOpen() const
```

Возвращаемое значение

Цена открытия позиции.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

StopLoss

Получает цену Stop Loss позиции.

```
double  StopLoss() const
```

Возвращаемое значение

Цена Stop Loss позиции.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TakeProfit

Получает цену Take Profit позиции.

```
double  TakeProfit() const
```

Возвращаемое значение

Цена Take Profit позиции.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PriceCurrent

Получает текущую цену по символу позиции.

```
double  PriceCurrent() const
```

Возвращаемое значение

Текущая цена по символу позиции.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Commission

Получает размер комиссии по позиции.

```
double  Commission() const
```

Возвращаемое значение

Размер комиссии по позиции в валюте депозита.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Swap

Получает размер свопа по позиции.

```
double  Swap() const
```

Возвращаемое значение

Размер свопа по позиции в валюте депозита.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Profit

Получает размер текущей прибыли по позиции.

```
double  Profit() const
```

Возвращаемое значение

Размер текущей прибыли по позиции в валюте депозита.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Symbol

Получает наименование символа позиции.

```
string  Symbol() const
```

Возвращаемое значение

Наименование символа позиции.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Comment

Получает комментарий позиции.

```
string  Comment() const
```

Возвращаемое значение

Комментарий позиции.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Получает значение указанного целочисленного свойства.

```
bool  InfoInteger(
   ENUM_POSITION_PROPERTY_INTEGER  prop_id,      // идентификатор свойства
   long&                           var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор целочисленного свойства из перечисления ENUM_POSITION_PROPERTY_INTEGER.

var

[out] Ссылка на переменную типа long для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Получает значение указанного double-свойства.

```
bool  InfoDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  prop_id,      // идентификатор свойства
   double&                        var          // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор double-свойства из перечисления ENUM_POSITION_PROPERTY_DOUBLE.

var

[out] Ссылка на переменную типа double для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoString

Получает значение указанного текстового свойства.

```
bool  InfoString(
   ENUM_POSITION_PROPERTY_STRING  prop_id,      // идентификатор свойства
   string&                        var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор текстового свойства из перечисления ENUM_POSITION_PROPERTY_STRING.

var

[out] Ссылка на переменную типа string для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Позиция должна быть предварительно выбрана для доступа методом Select (по символу) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Select

Выбирает позицию для дальнейшей работы.

```
bool  Select(
   const string  symbol      // символ
   )
```

Параметры

symbol

[in] Символ для выбора позиции.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

Выбирает позицию по индексу для дальнейшей работы.

```
bool  SelectByIndex(
   int  index      // индекс позиции
   )
```

Параметры

index

[in] Индекс позиции.

Возвращаемое значение

Возвращает true в случае успеха, или false, если позиция не выбрана.

© 2000-2025, MetaQuotes Ltd.

SelectByMagic

Выбирает позицию по имени инструмента и магическому номеру для дальнейшей работы.

```
bool  SelectByMagic(
   const string  symbol, // имя инструмента
   const ulong   magic   // магический номер
   );
```

Параметры

symbol

[in] Имя инструмента.

magic

[in] Магический номер позиции.

Возвращаемое значение

Возвращает true в случае успеха, или false, если позиция не выбрана.

© 2000-2025, MetaQuotes Ltd.

SelectByTicket

Выбирает позицию по тикету для дальнейшей работы.

```
bool  SelectByTicket(
   ulong  ticket      // тикет позиции
   )
```

Параметры

ticket

[in] Тикет позиции.

Возвращаемое значение

Возвращает true в случае успеха, или false, если позиция не выбрана.

© 2000-2025, MetaQuotes Ltd.

StoreState

Сохраняет параметры позиции.

```
void  StoreState()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

CheckState

Сравнивает текущие параметры с сохраненными.

```
bool  CheckState()
```

Возвращаемое значение

true - если параметры позиции изменились за время, прошедшее после последнего вызова метода StoreState(), иначе false.

© 2000-2025, MetaQuotes Ltd.

Класс CDealInfo

Класс CDealInfo является классом для упрощенного доступа к свойствам сделки.

Описание

Класс CDealInfo обеспечивает доступ к свойствам сделки.

Декларация

```
class CDealInfo : public CObject
```

Заголовок

```
#include <Trade\DealInfo.mqh>
```

Иерархия наследования

CObject

CDealInfo

Методы класса по группам

Доступ к целочисленным свойствам	
Order	Получает ордер, на основании которого выполнена сделка
Time	Получает время совершения сделки
TimeMsc	Получает время совершения сделки в миллисекундах с 01.01.1970
DealType	Получает тип сделки
TypeDescription	Получает тип сделки как строку
Entry	Получает направление сделки
EntryDescription	Получает направление сделки как строку
Magic	Получает идентификатор эксперта, совершившего сделку
PositionId	Получает идентификатор позиции, в которой участвовала сделка
Доступ к double-свойствам	
Volume	Получает объем сделки
Price	Получает цену сделки
Commision	Получает размер комиссии по сделке
Swap	Получает размер свопа при закрытии позиции
Profit	Получает финансовый результат сделки

© 2000-2025, MetaQuotes Ltd.

Доступ к целочисленным свойствам	
Доступ к текстовым свойствам	
Symbol	Получает наименование символа сделки
Comment	Получает комментарий к сделке
Доступ к функциям API MQL5	
InfoInteger	Получает значение указанного целочисленного свойства
InfoDouble	Получает значение указанного double-свойства
InfoString	Получает значение указанного текстового свойства
Выбор	
Ticket	Получает тикет/выбирает сделку
SelectByIndex	Выбирает сделку по индексу

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Order

Получает ордер, на основании которого выполнена сделка.

```
long  Order() const
```

Возвращаемое значение

Ордер, на основании которого выполнена сделка.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Time

Получает время совершения сделки.

```
datetime  Time() const
```

Возвращаемое значение

Время совершения сделки.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TimeMsc

Получает время совершения сделки в миллисекундах с 01.01.1970.

```
ulong  TimeMsc() const
```

Возвращаемое значение

Время совершения сделки в миллисекундах с 01.01.1970.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

DealType

Получает тип сделки.

```
ENUM_DEAL_TYPE  DealType() const
```

Возвращаемое значение

Тип сделки из перечисления ENUM_DEAL_TYPE.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

TypeDescription

Получает тип сделки как строку.

```
string  TypeDescription() const
```

Возвращаемое значение

Тип сделки как строка.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Entry

Сравнивает направление сделки.

```
ENUM_DEAL_ENTRY  Entry() const
```

Возвращаемое значение

Направление сделки (значение перечисления ENUM_DEAL_ENTRY).

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

EntryDescription

Получает направление сделки как строку.

```
string  EntryDescription() const
```

Возвращаемое значение

Направление сделки как строка.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Magic

Получает идентификатор эксперта, совершившего сделку.

```
long  Magic() const
```

Возвращаемое значение

Идентификатор эксперта, совершившего сделку.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

PositionId

Получает идентификатор позиции, в которой участвовала сделка.

```
long  PositionId() const
```

Возвращаемое значение

Идентификатор позиции, в которой участвовала сделка.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Volume

Получает объем сделки.

```
double  Volume() const
```

Возвращаемое значение

Объем сделки.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Price

Получает цену сделки.

```
double  Price() const
```

Возвращаемое значение

Цена сделки.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Commission

Получает размер комиссии по сделке.

```
double  Commission() const
```

Возвращаемое значение

Размер комиссии по сделке.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Swap

Получает размер свопа при закрытии позиции.

```
double  Swap() const
```

Возвращаемое значение

Размер свопа при закрытии позиции.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Profit

Получает финансовый результат сделки.

```
double  Profit() const
```

Возвращаемое значение

Финансовый результат сделки в валюте депозита.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Symbol

Получает наименование символа сделки.

```
string  Symbol() const
```

Возвращаемое значение

Наименование символа сделки.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Comment

Получает комментарий к сделке.

```
string  Comment() const
```

Возвращаемое значение

Комментарий к сделке.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Получает значение указанного целочисленного свойства.

```
bool  InfoInteger(
   ENUM_DEAL_PROPERTY_INTEGER  prop_id,      // идентификатор свойства
   long&                       var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор целочисленного свойства из перечисления ENUM_DEAL_PROPERTY_INTEGER.

var

[out] Ссылка на переменную типа long для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoDouble

Получает значение указанного double-свойства.

```
bool  InfoDouble(
   ENUM_DEAL_PROPERTY_DOUBLE  prop_id,      // идентификатор свойства
   double&                    var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор double-свойства из перечисления ENUM_DEAL_PROPERTY_DOUBLE.

var

[in] Ссылка на переменную типа double для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

InfoString

Получает значение указанного текстового свойства.

```
bool  InfoString(
   ENUM_DEAL_PROPERTY_STRING  prop_id,      // идентификатор свойства
   string&                    var           // ссылка на переменную
   ) const
```

Параметры

prop_id

[in] Идентификатор текстового свойства из перечисления ENUM_DEAL_PROPERTY_STRING.

var

[out] Ссылка на переменную типа string для размещения результата.

Возвращаемое значение

true - в случае удачи, false - если не удалось получить значение свойства.

Примечание

Сделка должна быть предварительно выбрана для доступа методом Ticket (по тикету) или SelectByIndex (по индексу).

© 2000-2025, MetaQuotes Ltd.

Ticket (метод Get)

Получает тикет сделки.

```
ulong  Ticket() const
```

Возвращаемое значение

Тикет сделки.

Ticket (метод Set)

Выбирает сделку для дальнейшей работы.

```
void  Ticket(
   ulong  ticket      // тикет
   )
```

Параметры

ticket

[in] Тикет сделки.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

Выбирает сделку по индексу для дальнейшей работы.

```
bool  SelectByIndex(
   int  index      // индекс сделки
   )
```

Параметры

index

[in] Индекс сделки.

Возвращаемое значение

Возвращает true в случае успеха, или false, если сделка не выбрана.

© 2000-2025, MetaQuotes Ltd.

Класс CTrade

Класс CTrade является классом для упрощенного доступа к торговым функциям.

Описание

Класс CTrade обеспечивает упрощенный доступ к торговым функциям.

Декларация

```
class CTrade : public CObject
```

Заголовок

```
#include <Trade\Trade.mqh>
```

Иерархия наследования

CObject

CTrade

Прямые потомки

CExpertTrade

Методы класса по группам

Установка параметров	
LogLevel	Устанавливает уровень логирования сообщений
SetExpertMagicNumber	Устанавливает идентификатор эксперта
SetDeviationInPoints	Устанавливает допустимое проскальзывание
SetTypeFilling	Устанавливает тип ордера по исполнению
SetTypeFillingBySymbol	Устанавливает тип ордера по исполнению согласно настройкам указанного символа
SetAsyncMode	Устанавливает асинхронный режим торговых операций
SetMarginMode	Устанавливает режим расчета маржи в соответствии с настройками текущего счета
Операции с ордерами	
OrderOpen	Размещает отложенный ордер с заданными параметрами
OrderModify	Изменяет параметры отложенного ордера
OrderDelete	Удаляет отложенный ордер
Операции с позициями	
PositionOpen	Открывает позицию с заданными параметрами

© 2000-2025, MetaQuotes Ltd.

Установка параметров	
PositionModify	Изменяет параметры позиции по указанному символу или тикету позиции
PositionClose	Закрывает позицию по указанному символу
PositionClosePartial	Закрывает часть позиции по указанному символу или с указанным тикетом
PositionCloseBy	Закрывает позицию с указанным тикетом позицией встречного направления
Дополнительные методы	
Buy	Открывает длинную позицию с заданными параметрами
Sell	Открывает короткую позицию с заданными параметрами
BuyLimit	Устанавливает отложенный ордер на покупку по цене лучше рыночной
BuyStop	Устанавливает отложенный ордер на покупку по цене хуже рыночной
SellLimit	Устанавливает отложенный ордер на продажу по цене лучше рыночной
SellStop	Устанавливает отложенный ордер на продажу по цене хуже рыночной
Доступ к параметрам последнего запроса	
Request	Получает копию структуры последнего запроса
RequestAction	Получает тип торговой операции
RequestActionDescription	Получает тип торговой операции как строку
RequestMagic	Получает идентификатор эксперта
RequestOrder	Получает тикет ордера
RequestSymbol	Получает имя торгового инструмента, по которому выставляется ордер
RequestVolume	Получает запрашиваемый объем сделки в лотах
RequestPrice	Получает цену, при достижении которой ордер должен быть исполнен
RequestStopLimit	Получает цену, по которой будет выставлен отложенный Stop Limit ордер
RequestSL	Получает цену, по которой сработает Stop Loss
RequestTP	Получает цену, по которой сработает Take Profit

© 2000-2025, MetaQuotes Ltd.

Установка параметров	
RequestDeviation	Получает максимально приемлемое отклонение от запрашиваемой цены
RequestType	Получает тип ордера
RequestTypeDescription	Получает тип ордера как строку
RequestTypeFilling	Получает тип ордера по исполнению
RequestTypeFillingDescription	Получает тип ордера по исполнению как строку
RequestTypeTime	Получает тип ордера по истечению
RequestTypeTimeDescription	Получает тип ордера по истечению как строку
RequestExpiration	Получает срок истечения отложенного ордера
RequestComment	Получает комментарий к ордеру
RequestPosition	Получает тикет позиции
RequestPositionBy	Получает тикет встречной позиции
Доступ к результатам проверки последнего запроса	
CheckResult	Получает копию структуры результатов проверки последнего запроса
CheckResultRetcode	Возвращает значение поля retcode структуры MqlTradeCheckResult, заполненной при проверке правильности запроса
CheckResultRetcodeDescription	Возвращает строковое описание поля retcode структуры MqlTradeCheckResult, заполненной при проверке правильности запроса
CheckResultBalance	Возвращает значение поля balance структуры MqlTradeCheckResult, заполненной при проверке правильности запроса
CheckResultEquity	Возвращает значение поля equity структуры MqlTradeCheckResult, заполненной при проверке правильности запроса
CheckResultProfit	Получает значение плавающей прибыли, которая будет после выполнения торговой операции
CheckResultMargin	Возвращает значение поля margin структуры MqlTradeCheckResult, заполненной при проверке правильности запроса
CheckResultMarginFree	Возвращает значение поля margin_free структуры MqlTradeCheckResult, заполненной при проверке правильности запроса

© 2000-2025, MetaQuotes Ltd.

Установка параметров	
CheckResultMarginLevel	Возвращает значение поля margin_level структуры MqlTradeCheckResult, заполненной при проверке правильности запроса
CheckResultComment	Возвращает значение поля comment структуры MqlTradeCheckResult, заполненной при проверке правильности запроса
Доступ к результатам исполнения последнего запроса	
Result	Получает копию структуры результатов исполнения последнего запроса
ResultRetcode	Получает код результата выполнения запроса
ResultRetcodeDescription	Получает код результата выполнения запроса как строку
ResultDeal	Получает тикет сделки
ResultOrder	Получает тикет ордера
ResultVolume	Получает объем сделки или ордера
ResultPrice	Получает цену, подтвержденную брокером
ResultBid	Получает текущую цену предложения (реквоту)
ResultAsk	Получает текущую цену спроса (реквоту)
ResultComment	Получает комментарий брокера
Вспомогательные методы	
PrintRequest	Выводит в журнал параметры последнего запроса
PrintResult	Выводит в журнал результаты исполнения последнего запроса
FormatRequest	Выводит в строку параметры последнего запроса
FormatRequestResult	Выводит в строку результаты исполнения последнего запроса

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

LogLevel

Устанавливает уровень логирования сообщений.

```
void  LogLevel(
   ENUM_LOG_LEVELS  log_level      // уровень
   )
```

Параметры

log_level

[in] Новый уровень логирования.

Возвращаемое значение

Нет.

Примечание

LOG_LEVEL_NO и меньше, отключает вывод любых сообщений (автоматически устанавливается в режиме оптимизации). LOG_LEVEL_ERRORS включает вывод только сообщений об ошибках (значение по умолчанию). LOG_LEVEL_ALL и больше, включает вывод любых сообщений (автоматически устанавливается в режиме тестирования).

ENUM_LOG_LEVELS

Идентификатор	Описание	Значение
LOG_LEVEL_NO	Вывод сообщений отключен	0
LOG_LEVEL_ERRORS	Выводятся только сообщения об ошибках	1
LOG_LEVEL_ALL	Выводятся все сообщения	2

© 2000-2025, MetaQuotes Ltd.

SetExpertMagicNumber

Устанавливает идентификатор эксперта

```
void  SetExpertMagicNumber(
   ulong  magic      // идентификатор
   )
```

Параметры

magic

[in] Новый идентификатор эксперта.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

SetDeviationInPoints

Устанавливает допустимое проскальзывание.

```
void  SetDeviationInPoints(
   ulong  deviation      // проскальзывание
   )
```

Параметры

deviation

[in] Допустимое проскальзывание в пунктах.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

SetTypeFilling

Устанавливает тип ордера по исполнению.

```
void  SetTypeFilling(
   ENUM_ORDER_TYPE_FILLING  filling      // тип ордера по исполнению
   )
```

Параметры

filling

[in] Тип ордера по исполнению из перечисления ENUM_ORDER_TYPE_FILLING.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

SetTypeFillingBySymbol

Устанавливает тип ордера по исполнению согласно настройкам указанного символа.

```
bool  SetTypeFillingBySymbol(
   const string   symbol      // имя финансового инструмента
   )
```

Параметры

symbol

[in] Имя символа, в котором SYMBOL_FILLING_MODE содержит допустимые политики исполнения ордеров.

Возвращаемое значение

true - в случае удачного выполнения, false - если не удалось установить политику исполнения.

Примечание

Если для символа одновременно разрешены политики исполнения SYMBOL_FILLING_FOK и SYMBOL_FILLING_IOC, то для ордера будет установлено значение ORDER_FILLING_FOK.

© 2000-2025, MetaQuotes Ltd.

SetAsyncMode

Устанавливает асинхронный режим проведения торговых операций.

```
void  SetAsyncMode(
   bool  mode      // асинхронный режим торговых операций
   )
```

Параметры

mode

[in] Статус использования асинхронного режима торговых операций.

Возвращаемое значение

Нет.

Примечание

Асинхронный режим позволяет проводить торговые операции без ожидания ответа торгового сервера (см. OrderSendAsync).

© 2000-2025, MetaQuotes Ltd.

SetMarginMode

Устанавливает режим расчета маржи в соответствии с настройками текущего счета.

```
void  SetMarginMode()
```

Возвращаемое значение

Нет.

Примечание

Режим расчета маржи указывается в ENUM_ACCOUNT_MARGIN_MODE.

© 2000-2025, MetaQuotes Ltd.

OrderOpen

Размещает отложенный ордер с заданными параметрами.

```
bool  OrderOpen(
   const string           symbol,           // символ
   ENUM_ORDER_TYPE        order_type,       // тип ордера
   double                 volume,           // объем ордера
   double                 limit_price,      // цена стоплимита
   double                 price,            // цена исполнения
   double                 sl,               // цена stop loss
   double                 tp,               // цена take profit
   ENUM_ORDER_TYPE_TIME   type_time,        // тип по истечению
   datetime               expiration,       // истечение
   const string           comment=""        // комментарий
   )
```

Параметры

symbol

[in] Наименование торгового инструмента.

order_type

[in] Тип торговой операции для ордера из перечисления ENUM_ORDER_TYPE.

volume

[in] Запрашиваемый объем ордера.

limit_price

[in] Цена, по которой будет выставлен отложенный Stop Limit ордер.

price

[in] Цена, по которой ордер должен быть исполнен.

sl

[in] Цена, по которой сработает Stop Loss (ограничение потерь).

tp

[in] Цена, по которой сработает Take Profit (фиксация прибыли).

type_time

[in] Тип ордера по истечению из перечисления ENUM_ORDER_TYPE_TIME.

expiration

[in] Срок истечения отложенного ордера.

comment=""

[in] Комментарий к ордеру.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

© 2000-2025, MetaQuotes Ltd.

Примечание

Успешное окончание работы метода OrderOpen(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode(), а также значение, возвращаемое методом ResultOrder().

© 2000-2025, MetaQuotes Ltd.

OrderModify

Изменяет параметры отложенного ордера.

```
bool  OrderModify(
   ulong                 ticket,          // тикет ордера
   double                price,           // цена исполнения
   double                sl,              // цена stop loss
   double                tp,              // цена take profit
   ENUM_ORDER_TYPE_TIME  type_time,       // тип по истечению
   datetime              expiration,      // истечение
   double                stoplimit        // цена Limit ордера
   )
```

Параметры

ticket

[in] Тикет ордера.

price

[in] Новая цена, по которой ордер должен быть исполнен (либо предыдущее значение, если изменение не нужно).

sl

[in] Новая цена, по которой сработает Stop Loss (либо предыдущее значение, если изменение не нужно).

tp

[in] Новая цена, по которой сработает Take Profit (либо предыдущее значение, если изменение не нужно).

type_time

[in] Новый тип ордера по истечению из перечисления ENUM_ORDER_TYPE_TIME (либо предыдущее значение, если изменение не нужно).

expiration

[in] Новый срок истечения отложенного ордера (либо предыдущее значение, если изменение не нужно).

stoplimit

[in] Новая цена, по которой будет выставлен Limit ордер при достижении ценой значения *price*. Указывается только для StopLimit ордеров.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода OrderModify(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode().

© 2000-2025, MetaQuotes Ltd.

OrderDelete

Удаляет отложенный ордер.

```
bool  OrderDelete(
   ulong  ticket      // тикет ордера
   )
```

Параметры

ticket

[in] Тикет ордера.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода OrderDelete(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode().

© 2000-2025, MetaQuotes Ltd.

PositionOpen

Открывает позицию с заданными параметрами по указанному символу.

```
bool  PositionOpen(
   const string      symbol,          // символ
   ENUM_ORDER_TYPE   order_type,      // тип торговой операции для открытия позиции
   double            volume,          // объем позиции
   double            price,           // цена исполнения
   double            sl,              // цена Stop Loss
   double            tp,              // цена Take Profit
   const string      comment=""       // комментарий
   )
```

Параметры

symbol

[in] Наименование торгового инструмента, по которому предполагается открыть позицию.

order_type

[in] Тип торговой операции для открытия позиции из перечисления ENUM_ORDER_TYPE.

volume

[in] Запрашиваемый объем позиции.

price

[in] Цена, по которой позиция должна быть открыта.

sl

[in] Цена, по которой сработает Stop Loss (ограничение потерь).

tp

[in] Цена, по которой сработает Take Profit (фиксация прибыли).

comment=""

[in] Комментарий к позиции.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода PositionOpen(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode(), а также значение, возвращаемое методом ResultDeal().

© 2000-2025, MetaQuotes Ltd.

PositionModify

Изменяет параметры позиции по указанному символу.

```
bool  PositionModify(
   const string  symbol,      // символ
   double        sl,          // цена Stop Loss
   double        tp           // цена Take Profit
   )
```

Изменяет параметры позиции по указанному тикету.

```
bool  PositionModify(
   const ulong   ticket,      // тикет позиции
   double        sl,          // цена Stop Loss
   double        tp           // цена Take Profit
   )
```

Параметры

symbol

[in] Наименование торгового инструмента, по которому предполагается модифицировать позицию.

ticket

[in] Тикет позиции, которую предполагается модифицировать.

sl

[in] Новая цена, по которой сработает Stop Loss (либо, если изменение не нужно, предыдущее значение).

tp

[in] Новая цена, по которой сработает Take Profit (либо, если изменение не нужно, предыдущее значение).

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода PositionModify(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode().

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций. В этом случае, PositionModify изменит позицию с наименьшим тикетом.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

PositionClose

Закрывает позицию по указанному символу.

```
bool  PositionClose(
   const string  symbol,                  // символ
   ulong         deviation=ULONG_MAX      // отклонение
   )
```

Закрывает позицию с указанным тикетом.

```
bool  PositionClose(
   const ulong   ticket,                  // тикет позиции
   ulong         deviation=ULONG_MAX      // отклонение
   )
```

Параметры

symbol

[in] Наименование торгового инструмента, по которому предполагается закрыть позицию.

ticket

[in] Тикет закрываемой позиции.

deviation=ULONG_MAX

[in] Максимальное отклонение от текущей цены (в пунктах).

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода PositionClose(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode().

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций. В этом случае, PositionClose закроет позицию с наименьшим тикетом.

© 2000-2025, MetaQuotes Ltd.

PositionClosePartial

Закрывает часть позиции по указанному символу при "хеджинговом" учете.

```
bool  PositionClosePartial(
   const string  symbol,                  // символ
   const double  volume,                  // объем
   ulong         deviation=ULONG_MAX      // отклонение
   )
```

Закрывает часть позиции с указанным тикетом при "хеджинговом" учете.

```
bool  PositionClosePartial(
   const ulong   ticket,                  // тикет позиции
   const double  volume,                  // объем
   ulong         deviation=ULONG_MAX      // отклонение
   )
```

Параметры

symbol

[in] Наименование торгового инструмента, по которому предполагается закрыть часть позиции. Если для частичного закрытия позиции указан символ (не тикет), то будет выбрана первая найденная позиция по данному символу, имеющая заданный MagicNumber (идентификатор эксперта). Поэтому в ряде случаев лучше использовать вариант PositionClosePartial() с указанием тикета позиции.

volume

[in] Объем, на который нужно уменьшить позицию. Если значение больше, чем объём частично закрываемой позиции, то произойдет полное закрытие позиции. Новая позиция в обратном направлении открыта не будет.

ticket

[in] Тикет закрываемой позиции.

deviation=ULONG_MAX

[in] Максимальное отклонение от текущей цены (в пунктах).

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода PositionClosePartial(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode().

При "неттинговом" учете позиций (ACCOUNT_MARGIN_MODE_RETAIL_NETTING и ACCOUNT_MARGIN_MODE_EXCHANGE) по каждому символу в любой момент времени может быть открыта только одна позиция, которая является результатом одной или более сделок. Не следует путать между собой позиции и действующие отложенные ордера, которые также отображаются на вкладке "Торговля" в панели "Инструменты".

© 2000-2025, MetaQuotes Ltd.

При независимом представлении позиций (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) по каждому символу одновременно может быть открыто несколько позиций. В этом случае, PositionClose закроет позицию с наименьшим тикетом.

© 2000-2025, MetaQuotes Ltd.

PositionCloseBy

Закрывает позицию с указанным тикетом позицией встречного направления.

```
bool  PositionCloseBy(
   const ulong   ticket,        // тикет позиции
   const ulong   ticket_by      // тикет встречной позции
   )
```

Параметры

ticket

[in] Тикет закрываемой позиции.

ticket_by

[in] Тикет встречной позиции, которая используется для закрытия.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода PositionCloseBy(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode().

© 2000-2025, MetaQuotes Ltd.

Buy

Открывает длинную позицию с заданными параметрами.

```
bool  Buy(
   double         volume,           // объем позиции
   const string   symbol=NULL,      // символ
   double         price=0.0,        // цена исполнения
   double         sl=0.0,           // цена Stop Loss
   double         tp=0.0,           // цена Take Profit
   const string   comment=""        // комментарий
   )
```

Параметры

volume

[in] Запрашиваемый объем позиции.

symbol=NULL

[in] Наименование торгового инструмента, по которому предполагается открыть позицию. Если символ не указан, то позиция откроется по текущему инструменту.

price=0.0

[in] Цена, по которой позиция должна быть открыта. Если цена не указана, то позиция откроется по текущей рыночной цене Ask.

sl=0.0

[in] Цена, по которой сработает Stop Loss (ограничение потерь).

tp=0.0

[in] Цена, по которой сработает Take Profit (фиксация прибыли).

comment=""

[in] Комментарий к позиции.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода Buy(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode(), а также значение, возвращаемое методом ResultDeal().

© 2000-2025, MetaQuotes Ltd.

Sell

Открывает короткую позицию с заданными параметрами.

```
bool  Sell(
   double         volume,           // объем позиции
   const string   symbol=NULL,      // символ
   double         price=0.0,        // цена исполнения
   double         sl=0.0,           // цена Stop Loss
   double         tp=0.0,           // цена Take Profit
   const string   comment=""        // комментарий
   )
```

Параметры

volume

[in] Запрашиваемый объем позиции.

symbol=NULL

[in] Наименование торгового инструмента, по которому предполагается открыть позицию. Если символ не указан, то позиция откроется по текущему инструменту.

price=0.0

[in] Цена, по которой позиция должна быть открыта. Если цена не указана, то позиция откроется по текущей рыночной цене Bid.

sl=0.0

[in] Цена, по которой сработает Stop Loss (ограничение потерь).

tp=0.0

[in] Цена, по которой сработает Take Profit (фиксация прибыли).

comment=""

[in] Комментарий к позиции.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода Sell(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode(), а также значение, возвращаемое методом ResultDeal().

© 2000-2025, MetaQuotes Ltd.

BuyLimit

Устанавливает отложенный ордер на покупку по цене лучше рыночной.

```
bool  BuyLimit(
   double                volume,                    // объем позиции
   double                price,                     // цена исполнения
   const string          symbol=NULL,               // символ
   double                sl=0.0,                    // цена Stop Loss
   double                tp=0.0,                    // цена Take Profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // тип истечения
   datetime              expiration=0,              // время истечения
   const string          comment=""                 // комментарий
   )
```

Параметры

volume

[in] Запрашиваемый объем ордера.

price

[in] Цена, по которой предполагается установить ордер.

symbol=NULL

[in] Наименование торгового инструмента, по которому предполагается установить ордер. Если символ не указан, то ордер установится по текущему инструменту.

sl=0.0

[in] Цена, по которой сработает Stop Loss (ограничение потерь).

tp=0.0

[in] Цена, по которой сработает Take Profit (фиксация прибыли).

type_time=ORDER_TIME_GTC

[in] Тип истечения срока действия ордера из перечисления ENUM_ORDER_TYPE_TIME.

expiration=0

[in] Время истечения срока действия ордера (только для type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Комментарий к ордеру.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода BuyLimit(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode(), а также значение, возвращаемое методом ResultOrder().

© 2000-2025, MetaQuotes Ltd.

BuyStop

Устанавливает отложенный ордер на покупку по цене хуже рыночной.

```
bool  BuyStop(
   double                volume,                    // объем позиции
   double                price,                     // цена исполнения
   const string          symbol=NULL,               // символ
   double                sl=0.0,                    // цена Stop Loss
   double                tp=0.0,                    // цена Take Profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // тип истечения
   datetime              expiration=0,              // время истечения
   const string          comment=""                 // комментарий
   )
```

Параметры

volume

[in] Запрашиваемый объем ордера.

price

[in] Цена, по которой предполагается установить ордер.

symbol=NULL

[in] Наименование торгового инструмента, по которому предполагается установить ордер. Если символ не указан, то ордер установится по текущему инструменту.

sl=0.0

[in] Цена, по которой сработает Stop Loss (ограничение потерь).

tp=0.0

[in] Цена, по которой сработает Take Profit (фиксация прибыли).

type_time=ORDER_TIME_GTC

[in] Тип истечения срока действия ордера из перечисления ENUM_ORDER_TYPE_TIME.

expiration=0

[in] Время истечения срока действия ордера (только для type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Комментарий к ордеру.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода BuyStop(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode(), а также значение, возвращаемое методом ResultOrder().

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

SellLimit

Устанавливает отложенный ордер на продажу по цене лучше рыночной.

```
bool  SellLimit(
   double                volume,                    // объем позиции
   double                price,                     // цена исполнения
   const string          symbol=NULL,               // символ
   double                sl=0.0,                    // цена Stop Loss
   double                tp=0.0,                    // цена Take Profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // тип истечения
   datetime              expiration=0,              // время истечения
   const string          comment=""                 // комментарий
   )
```

Параметры

volume

[in] Запрашиваемый объем ордера.

price

[in] Цена, по которой предполагается установить ордер.

symbol=NULL

[in] Наименование торгового инструмента, по которому предполагается установить ордер. Если символ не указан, то ордер установится по текущему инструменту.

sl=0.0

[in] Цена, по которой сработает Stop Loss (ограничение потерь).

tp=0.0

[in] Цена, по которой сработает Take Profit (фиксация прибыли).

type_time=ORDER_TIME_GTC

[in] Тип истечения срока действия ордера из перечисления ENUM_ORDER_TYPE_TIME.

expiration=0

[in] Время истечения срока действия ордера (только для type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Комментарий к ордеру.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода SellLimit(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode(), а также значение, возвращаемое методом ResultOrder().

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

SellStop

Устанавливает отложенный ордер на продажу по цене хуже рыночной.

```
bool  SellStop(
   double                volume,                    // объем позиции
   double                price,                     // цена исполнения
   const string          symbol=NULL,               // символ
   double                sl=0.0,                    // цена Stop Loss
   double                tp=0.0,                    // цена Take Profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // тип истечения
   datetime              expiration=0,              // время истечения
   const string          comment=""                 // комментарий
   )
```

Параметры

volume

[in] Запрашиваемый объем ордера.

price

[in] Цена, по которой предполагается установить ордер.

symbol=NULL

[in] Наименование торгового инструмента, по которому предполагается установить ордер. Если символ не указан, то ордер установится по текущему инструменту.

sl=0.0

[in] Цена, по которой сработает Stop Loss (ограничение потерь).

tp=0.0

[in] Цена, по которой сработает Take Profit (фиксация прибыли).

type_time=ORDER_TIME_GTC

[in] Тип истечения срока действия ордера из перечисления ENUM_ORDER_TYPE_TIME.

expiration=0

[in] Время истечения срока действия ордера (только для type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Комментарий к ордеру.

Возвращаемое значение

true - в случае успешной базовой проверки структур, иначе false.

Примечание

Успешное окончание работы метода SellStop(...) не всегда означает успешное совершение торговой операции. Необходимо проверять результат выполнения торгового запроса (код возврата торгового сервера) вызовом метода ResultRetcode(), а также значение, возвращаемое методом ResultOrder().

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Request

Получает копию структуры последнего запроса.

```
void  Request(
   MqlTradeRequest&  request      // ссылка
   ) const
```

Параметры

request

[out] Ссылка на структуру для копирования.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

RequestAction

Получает тип торговой операции.

```
ENUM_TRADE_REQUEST_ACTIONS  RequestAction() const
```

Возвращаемое значение

Тип торговой операции, использованный в последнем запросе из перечисления ENUM_TRADE_REQUEST_ACTIONS.

© 2000-2025, MetaQuotes Ltd.

RequestActionDescription

Получает тип торговой операции как строку.

```
string  RequestActionDescription()) const
```

Возвращаемое значение

Тип торговой операции, использованный в последнем запросе, как строку.

© 2000-2025, MetaQuotes Ltd.

RequestMagic

Получает идентификатор эксперта.

```
ulong  RequestMagic() const
```

Возвращаемое значение

Идентификатор эксперта, использованный в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestOrder

Получает тикет ордера.

```
ulong  RequestOrder() const
```

Возвращаемое значение

Тикет ордера, использованный в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestSymbol

Получает имя торгового инструмента.

```
string  RequestSymbol() const
```

Возвращаемое значение

Имя торгового инструмента, по которому выставляется ордер, использованное в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestVolume

Получает запрашиваемый объем сделки в лотах.

```
double  RequestVolume() const
```

Возвращаемое значение

Запрашиваемый объем сделки в лотах, использованный в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestPrice

Получает цену, при достижении которой ордер должен быть исполнен.

```
double  RequestPrice() const
```

Возвращаемое значение

Цена, использованная в последнем запросе, при достижении которой ордер должен быть исполнен.

© 2000-2025, MetaQuotes Ltd.

RequestStopLimit

Получает цену, по которой будет выставлен отложенный Stop Limit ордер.

```
double  RequestStopLimit() const
```

Возвращаемое значение

Цена, использованная в последнем запросе, по которой будет выставлен отложенный Stop Limit ордер.

© 2000-2025, MetaQuotes Ltd.

RequestSL

Получает цену, по которой сработает Stop Loss.

```
double  RequestSL() const
```

Возвращаемое значение

Цена, использованная в последнем запросе, по которой сработает Stop Loss.

© 2000-2025, MetaQuotes Ltd.

RequestTP

Получает цену, по которой сработает Take Profit.

```
double  RequestTP() const
```

Возвращаемое значение

Цена, использованная в последнем запросе, по которой сработает Take Profit.

© 2000-2025, MetaQuotes Ltd.

RequestDeviation

Получает максимальное отклонение от запрашиваемой цены.

```
ulong  RequestDeviation() const
```

Возвращаемое значение

Максимально приемлемое отклонение от запрашиваемой цены, использованное в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestType

Получает тип ордера.

```
ENUM_ORDER_TYPE  RequestType() const
```

Возвращаемое значение

Тип ордера, использованный в последнем запросе из перечисления ENUM_ORDER_TYPE.

© 2000-2025, MetaQuotes Ltd.

RequestTypeTimeDescription

Получает тип ордера по истечению как строку.

```
string  RequestTypeTimeDescription() const
```

Возвращаемое значение

Тип ордера по истечению, использованный в последнем запросе, как строку.

© 2000-2025, MetaQuotes Ltd.

RequestTypeFilling

Получает тип ордера по исполнению.

```
ENUM_ORDER_TYPE_FILLING  RequestTypeFilling() const
```

Возвращаемое значение

Тип ордера по исполнению (из перечисления ENUM_ORDER_TYPE_FILLING), использованный в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestTypeFillingDescription

Получает тип ордера по исполнению как строку.

```
string  RequestTypeFillingDescription() const
```

Возвращаемое значение

Тип ордера по исполнению, использованный в последнем запросе, как строку.

© 2000-2025, MetaQuotes Ltd.

RequestTypeTime

Получает тип ордера по истечению.

```
ENUM_ORDER_TYPE_TIME  RequestTypeTime() const
```

Возвращаемое значение

Тип ордера по истечению (из перечисления ENUM_ORDER_TYPE_TIME), использованный в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestTypeTimeDescription

Получает тип ордера по истечению как строку.

```
string  RequestTypeTimeDescription() const
```

Возвращаемое значение

Тип ордера по истечению, использованный в последнем запросе, как строку.

© 2000-2025, MetaQuotes Ltd.

RequestExpiration

Получает срок истечения отложенного ордера.

```
datetime  RequestExpiration() const
```

Возвращаемое значение

Срок истечения отложенного ордера, использованный в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestComment

Получает комментарий к ордеру.

```
string  RequestComment() const
```

Возвращаемое значение

Комментарий к ордеру, использованный в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestPosition

Получает тикет позиции.

```
ulong  RequestPosition() const
```

Возвращаемое значение

Тикет позиции, использованный в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

RequestPositionBy

Получает тикет встречной позиции.

```
ulong  RequestPositionBy() const
```

Возвращаемое значение

Тикет встречной позиции, использованный в последнем запросе.

© 2000-2025, MetaQuotes Ltd.

Result

Получает копию структуры результата выполнения последнего запроса.

```
void  Result(
   MqlTradeResult&  result      // ссылка
   ) const
```

Параметры

result

[out] Ссылка на структуру типа MqlTradeResult для копирования.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

ResultRetcode

Получает код результата выполнения запроса.

```
uint  ResultRetcode() const
```

Возвращаемое значение

Код результата последнего запроса.

© 2000-2025, MetaQuotes Ltd.

ResultRetcodeDescription

Получает код результата выполнения запроса как строку.

```
string  ResultRetcodeDescription() const
```

Возвращаемое значение

Возвращает строку с описанием результата последнего запроса.

© 2000-2025, MetaQuotes Ltd.

ResultDeal

Получает тикет сделки.

```
ulong  ResultDeal() const
```

Возвращаемое значение

Тикет сделки, если она совершена.

© 2000-2025, MetaQuotes Ltd.

ResultOrder

Получает тикет ордера.

```
ulong  ResultOrder() const
```

Возвращаемое значение

Тикет ордера, если он выставлен.

© 2000-2025, MetaQuotes Ltd.

ResultVolume

Получает объем сделки или ордера.

```
double  ResultVolume() const
```

Возвращаемое значение

Объем сделки или ордера, подтверждённый брокером.

© 2000-2025, MetaQuotes Ltd.

ResultPrice

Получает цену, подтвержденную брокером.

```
double  ResultPrice() const
```

Возвращаемое значение

Цена, подтвержденная брокером.

© 2000-2025, MetaQuotes Ltd.

ResultBid

Получает текущую цену предложения (реквоту).

```
double  ResultBid() const
```

Возвращаемое значение

Текущая рыночная цена предложения (реквота).

© 2000-2025, MetaQuotes Ltd.

ResultAsk

Получает текущую цену спроса (реквоту).

```
double  ResultAsk() const
```

Возвращаемое значение

Текущая рыночная цена спроса (цены реквота).

© 2000-2025, MetaQuotes Ltd.

ResultComment

Получает комментарий брокера.

```
string  ResultComment() const
```

Возвращаемое значение

Комментарий брокера к операции.

© 2000-2025, MetaQuotes Ltd.

CheckResult

Получает копию структуры результата проверки последнего запроса.

```
void  CheckResult(
   MqlTradeCheckResult&  check_result      // ссылка
   ) const
```

Параметры

check_result

[out] Ссылка на структуру типа MqlTradeCheckResult для копирования.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

CheckResultRetcode

Возвращает значение поля retcode структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

```
uint  CheckResultRetcode() const
```

Возвращаемое значение

Значение поля retcode (код ошибки) структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

© 2000-2025, MetaQuotes Ltd.

CheckResultRetcodeDescription

Возвращает строковое описание поля retcode структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

```
string  ResultRetcodeDescription() const
```

Возвращаемое значение

Строковое описание поля retcode (код ошибки) структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

© 2000-2025, MetaQuotes Ltd.

CheckResultBalance

Возвращает значение поля balance структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

```
double  CheckResultBalance() const
```

Возвращаемое значение

Значение поля balance (значение баланса, которое будет после выполнения торговой операции) структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

© 2000-2025, MetaQuotes Ltd.

CheckResultEquity

Возвращает значение поля equity структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

```
double  CheckResultEquity() const
```

Возвращаемое значение

Значение поля equity (значение собственных средств, которое будет после выполнения торговой операции) структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

© 2000-2025, MetaQuotes Ltd.

CheckResultProfit

Получает значение плавающей прибыли.

```
double  CheckResultProfit() const
```

Возвращаемое значение

Значение плавающей прибыли, которая будет после выполнения торговой операции.

© 2000-2025, MetaQuotes Ltd.

CheckResultMargin

Возвращает значение поля margin структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

```
double  CheckResultMargin() const
```

Возвращаемое значение

Значение поля margin (размер маржи необходимый для требуемой торговой операции) структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

© 2000-2025, MetaQuotes Ltd.

CheckResultMarginFree

Возвращает значение поля margin_free структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

```
double  CheckResultMarginFree() const
```

Возвращаемое значение

Значение поля margin_free (размер свободных собственных средств, которые останутся после выполнения требуемой торговой операции) структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

© 2000-2025, MetaQuotes Ltd.

CheckResultMarginLevel

Возвращает значение поля margin_level структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

```
double  CheckResultMarginLevel() const
```

Возвращаемое значение

Значение поля margin_level (уровень маржи, который установится после выполнения требуемой торговой операции) структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

© 2000-2025, MetaQuotes Ltd.

CheckResultComment

Возвращает значение поля comment структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

```
string  CheckResultComment() const
```

Возвращаемое значение

Значение поля comment (комментарий к коду ответа, описание ошибки) структуры MqlTradeCheckResult, заполненной при проверке правильности запроса.

© 2000-2025, MetaQuotes Ltd.

PrintRequest

Выводит в журнал параметры последнего запроса.

```
void  PrintRequest() const
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

PrintResult

Выводит в журнал результаты исполнения последнего запроса.

```
void  PrintResult() const
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

FormatRequest

Выводит в строку параметры последнего запроса.

```
string  FormatRequest(
   string&                 str,          // строка
   const MqlTradeRequest&  request       // запрос
   ) const
```

Параметры

str

[in] Ссылка на строку для размещения результата.

request

[in] Ссылка на структуру типа MqlTradeRequest, содержащую информацию о последнем запросе.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

FormatRequestResult

Выводит в строку результаты исполнения последнего запроса.

```
string  FormatRequestResult(
   string&                 str,          // строка
   const MqlTradeRequest&  request,      // запрос
   const MqlTradeResult&   result        // результат
   ) const
```

Параметры

str

[in] Ссылка на строку для размещения результата.

request

[in] Ссылка на структуру типа MqlTradeRequest, содержащую информацию о последнем запросе.

result

[in] Ссылка на структуру типа MqlTradeResult, содержащую информацию о результате исполнения последнего запроса.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Класс CTerminalInfo

Класс CTerminalInfo является классом для упрощенного доступа к свойствам окружения mql5-программы.

Описание

Класс CTerminalInfo обеспечивает упрощенный доступ к свойствам окружения mql5-программы.

Декларация

```
class CTerminalInfo : public CObject
```

Заголовок

```
#include <Trade\TerminalInfo.mqh>
```

Иерархия наследования

CObject

CTerminalInfo

Методы класса по группам

Методы доступа к параметрам типа integer клиентского терминала	
Build	Получает номер билда запущенного терминала
IsConnected	Получает информацию о наличии подключения к торговому серверу
IsDLLsAllowed	Получает информацию о разрешении на использование DLL
IsTradeAllowed	Получает информацию о разрешении на торговлю
IsEmailEnabled	Получает информацию о разрешении на отправку писем с использованием SMTP- сервера и логина, указанных в настройках терминала
IsFtpEnabled	Получает информацию о разрешении на отправку отчетов по FTP для логина и FTP-сервера, указанного в настройках
MaxBars	Получает максимальное количество баров на графике
CodePage	Получает информацию о номере кодовой страницы языка, установленного в терминале
CPUCores	Получает информацию о количестве ядер процессора, установленных в системе
MemoryPhysical	Получает информацию о размере физической памяти в системе (в Mb)

© 2000-2025, MetaQuotes Ltd.

Методы доступа к параметрам типа integer клиентского терминала	
MemoryTotal	Получает информацию о размере памяти, доступной процессу терминала/агента (в Mb)
MemoryAvailable	Получает информацию о размере свободной памяти, доступной процессу терминала/агента (в Mb)
MemoryUsed	Получает информацию о размере памяти, используемой процессом терминала/агента (в Mb)
IsX64	Получает информацию о типе клиентского терминала/агента
OpenCLSupport	Получает информацию о версии OpenCL, поддерживамой видеокартой
DiskSpace	Получает информацию об объеме свободной памяти на диске
Методы доступа к параметрам типа string клиентского терминала	
Language	Получает информацию о языке терминала
Name	Получает информацию об наименовании терминала
Company	Получает информацию о наименовании компании
Path	Получает информацию о папке, в которой запущен терминал
DataPath	Получает информацию о папке, в которой хранятся данные терминала
CommonDataPath	Получает информацию о папке данных всех клиентских терминалов, установленных на компьютере
Доступ к функциям API MQL5	
InfoInteger	Получает информацию о параметре типа integer терминала
InfoString	Получает информацию о параметре типа string терминала

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Build

Получает номер билда запущенного терминала.

```
int  CBuild() const
```

Возвращаемое значение

Номер билда запущенного клиентского терминала.

Примечание

Номер билда терминала определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_BUILD).

© 2000-2025, MetaQuotes Ltd.

IsConnected

Получает информацию о наличии подключения к торговому серверу.

```
bool  IsConnected() const
```

Возвращаемое значение

true, если терминал подключен к серверу, иначе false.

Примечание

Подключение к торговому серверу определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_CONNECTED).

© 2000-2025, MetaQuotes Ltd.

IsDLLsAllowed

Получает информацию о разрешении на использование DLL.

```
bool  IsDLLsAllowed() const
```

Возвращаемое значение

true, если использование DLL разрешено, иначе false.

Примечание

Разрешение на использование DLL определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_DLLS_ALLOWED).

© 2000-2025, MetaQuotes Ltd.

IsTradeAllowed

Получает информацию о разрешении на торговлю.

```
bool  IsTradeAllowed() const
```

Возвращаемое значение

true, если торговля разрешена, иначе false.

Примечание

Разрешение на торговлю определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_TRADE_ALLOWED).

© 2000-2025, MetaQuotes Ltd.

IsEmailEnabled

Получает информацию о разрешении на отправку писем с использованием SMTP-сервера и логина, указанных в настройках терминала.

```
bool  IsEmailEnabled() const
```

Возвращаемое значение

true, если отправка писем разрешена, иначе false.

Примечание

Разрешение на отправку писем определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_EMAIL_ENABLED).

© 2000-2025, MetaQuotes Ltd.

IsFtpEnabled

Получает информацию о разрешении на отправку отчетов по FTP для логина и FTP-сервера, указанного в настройках терминала.

```
bool  IsFtpEnabled() const
```

Возвращаемое значение

true, если отправка отчетов по FTP разрешена, иначе false.

Примечание

Разрешение на отправку отчетов определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_FTP_ENABLED).

© 2000-2025, MetaQuotes Ltd.

MaxBars

Получает максимальное количество баров на графике, указанное в настройках терминала.

```
int  MaxBars() const
```

Возвращаемое значение

Максимальное количество баров на графике.

Примечание

Максимальное количество баров на графике определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_MAXBARS).

© 2000-2025, MetaQuotes Ltd.

CodePage

Получает информацию о номере кодовой страницы языка, установленного в терминале.

```
int  CodePage() const
```

Возвращаемое значение

Номер кодовой страницы языка, установленного в клиентском терминале.

Примечание

Номер кодовой страницы языка определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_CODEPAGE).

© 2000-2025, MetaQuotes Ltd.

CPUCores

Получает информацию о количестве ядер процессора, установленных в системе.

```
int  CPUCores() const
```

Возвращаемое значение

Количество ядер процессора.

Примечание

Количество ядер процессора определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_CPU_CORES).

© 2000-2025, MetaQuotes Ltd.

MemoryPhysical

Получает информацию о размере физической памяти в системе (в Mb).

```
int  MemoryPhysical() const
```

Возвращаемое значение

Размер физической памяти.

Примечание

Размер физической памяти определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_MEMORY_PHYSICAL).

© 2000-2025, MetaQuotes Ltd.

MemoryTotal

Получает информацию о размере памяти, доступной процессу терминала/агента (в Mb).

```
int  MemoryTotal() const
```

Возвращаемое значение

Размер памяти, доступный процессу терминала (агента).

Примечание

Размер памяти, доступный процессу терминала (агента), определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_MEMORY_TOTAL).

© 2000-2025, MetaQuotes Ltd.

MemoryAvailable

Получает информацию о размере свободной памяти, доступной процессу терминала/агента (в Mb).

```
int  MemoryTotal() const
```

Возвращаемое значение

Размер свободной памяти, доступный процессу терминала (агента).

Примечание

Размер свободной памяти, доступный процессу терминала (агента), определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_MEMORY_TOTAL).

© 2000-2025, MetaQuotes Ltd.

MemoryUsed

Получает информацию о размере памяти, используемой процессом терминала/агента (в Mb).

```
int  MemoryUsed() const
```

Возвращаемое значение

Размер памяти, используемой процессом терминала (агента).

Примечание

Размер памяти, используемой процессом терминала (агента), определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_MEMORY_USED).

© 2000-2025, MetaQuotes Ltd.

IsX64

Получает информацию о типе клиентского терминала/агента.

```
bool  IsX64() const
```

Возвращаемое значение

true, если используется 64-битная версия терминала, иначе false.

Примечание

Тип терминала определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_X64).

© 2000-2025, MetaQuotes Ltd.

OpenCLSupport

Получает информацию о версии OpenCL, поддерживаемой видеокартой.

```
int  OpenCLSupport() const
```

Возвращаемое значение

Версия OpenCL в виде 0x00010002 = "1.2". Значение 0 означает, что OpenCL не поддерживается.

Примечание

Версия OpenCL определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_OPENCL_SUPPORT).

© 2000-2025, MetaQuotes Ltd.

DiskSpace

Получает информацию об объеме свободной памяти на диске для папки MQL5\Files терминала/агента (в Mb).

```
int  MDiskSpace() const
```

Возвращаемое значение

Размер свободной дисковой памяти.

Примечание

Размер свободной памяти на диске определяется при помощи функции TerminalInfoInteger() (свойство TERMINAL_DISK_SPACE).

© 2000-2025, MetaQuotes Ltd.

Language

Получает информацию о языке клиентского терминала.

```
string  Language() const
```

Возвращаемое значение

Язык клиентского терминала.

Примечание

Язык терминала определяется при помощи функции TerminalInfoString() (свойство TERMINAL_LANGUAGE).

© 2000-2025, MetaQuotes Ltd.

Name

Получает информацию об наименовании клиентского терминала.

```
string  Name() const
```

Возвращаемое значение

Наименование клиентского терминала.

Примечание

Наименование терминала определяется при помощи функции TerminalInfoString() (свойство TERMINAL_NAME).

© 2000-2025, MetaQuotes Ltd.

Company

Получает информацию о наименовании компании.

```
string  Company() const
```

Возвращаемое значение

Наименование компании.

Примечание

Наименование компании определяется при помощи функции TerminalInfoString() (свойство TERMINAL_COMPANY).

© 2000-2025, MetaQuotes Ltd.

Path

Получает информацию о папке, в которой запущен терминал.

```
string  Path() const
```

Возвращаемое значение

Папка клиентского терминала.

Примечание

Папка клиентского терминала определяется при помощи функции TerminalInfoString() (свойство TERMINAL_PATH).

© 2000-2025, MetaQuotes Ltd.

DataPath

Получает информацию о папке, в которой хранятся данные терминала.

```
string  DataPath() const
```

Возвращаемое значение

Папка данных клиентского терминала.

Примечание

Папка данных клиентского терминала определяется при помощи функции TerminalInfoString() (свойство TERMINAL_DATA_PATH).

© 2000-2025, MetaQuotes Ltd.

CommonDataPath

Получает информацию об общей папке данных всех клиентских терминалов, установленных на компьютере.

```
string  CommonDataPath() const
```

Возвращаемое значение

Общая папка данных всех установленных терминалов.

Примечание

Общая папка данных всех установленных терминалов определяется при помощи функции TerminalInfoString() (свойство COMMON_DATA_PATH).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

Возвращает значение соответствующего integer-свойства окружения mql5-программы.

```
int  InfoInteger(
   int  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Может быть одним из значений перечисления ENUM_TERMINAL_INFO_INTEGER.

Возвращаемое значение

Значение типа int.

Примечание

Значение свойства определяется при помощи функции TerminalInfoInteger().

© 2000-2025, MetaQuotes Ltd.

InfoString

Функция возвращает значение соответствующего свойства окружения mql5-программы. Свойство должно быть типа string.

```
string  InfoString(
   int  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Может быть одним из значений перечисления ENUM_TERMINAL_INFO_STRING.

Возвращаемое значение

Значение типа string.

Примечание

Значение свойства определяется при помощи функции TerminalInfoString().

© 2000-2025, MetaQuotes Ltd.

Набор классов для создания и проверки торговых стратегий

Этот раздел содержит технические детали работы с классами для создания и проверки торговых стратегий и описание соответствующих компонентов стандартной библиотеки MQL5.

Использование классов для создания и проверки торговых стратегий позволит сэкономить время при разработке (а особенно,проверке) торговых стратегий.

Стандартная библиотека MQL5 (в части набора классов для создания и проверки торговых стратегий) размещается в рабочем каталоге терминала в папке Include\Expert.

Базовые классы экспертов	Описание
CExpertBase	Базовый класс для классов торговых стратегий
CExpert	Базовый класс торговой стратегии
CExpertSignal	Базовый класс для создания генераторов торговых сигналов
CExpertTrailing	Базовый класс сопровождения открытых позиций
CExpertMoney	Базовый класс для реализации алгоритмов управления капиталом и рисками

Классы торговых сигналов	Описание
CSignalAC	Модуль сигналов на основе рыночных моделей индикатора Accelerator Oscillator.
CSignalAMA	Модуль сигналов на основе рыночных моделей индикатора Adaptive Moving Average.
CSignalAO	Модуль сигналов на основе рыночных моделей индикатора Awesome Oscillator.
CSignalBearsPower	Модуль сигналов на основе рыночных моделей осциллятора Bears Power.
CSignalBullsPower	Модуль сигналов на основе рыночных моделей осциллятора Bulls Power.
CSignalCCI	Модуль сигналов на основе рыночных моделей осциллятора Commodity Channel Index.
CSignalDeM	Модуль сигналов на основе рыночных моделей осциллятора DeMarker.
CSignalDEMA	Модуль сигналов на основе рыночных моделей индикатора Double Exponential Moving Average.
CSignalEnvelopes	Модуль сигналов на основе рыночных моделей индикатора Envelopes.

© 2000-2025, MetaQuotes Ltd.

Классы торговых сигналов	Описание
CSignalFrAMA	Модуль сигналов на основе рыночных моделей индикатора Fractal Adaptive Moving Average.
CSignalITF	Модуль фильтрации сигналов по времени.
CSignalMACD	Модуль сигналов на основе рыночных моделей осциллятора MACD.
CSignalMA	Модуль сигналов на основе рыночных моделей индикатора Moving Average.
CSignalSAR	Модуль сигналов на основе рыночных моделей индикатора Parabolic SAR.
CSignalRSI	Модуль сигналов на основе рыночных моделей осциллятора Relative Strength Index.
CSignalRVI	Модуль сигналов на основе рыночных моделей осциллятора Relative Vigor Index.
CSignalStoch	Модуль сигналов на основе рыночных моделей осциллятора Stochastic.
CSignalTRIX	Модуль сигналов на основе рыночных моделей осциллятора Triple Exponential Average.
CSignalTEMA	Модуль сигналов на основе рыночных моделей индикатора Triple Exponential Moving Average.
CSignalWPR	Модуль сигналов на основе рыночных моделей осциллятора Williams Percent Range.

Классы сопровождения открытых позиций	Описание
CTrailingFixedPips	Класс реализации алгоритма сопровождения открытых позиций на фиксированном "расстоянии"
CTrailingMA	Класс реализации алгоритма сопровождения открытых позиций по значениям скользящей средней
CTrailingNone	Класс-заглушка алгоритмов сопровождения открытых позиций
CTrailingPSAR	Класс реализации алгоритма сопровождения открытых позиций по значениям индикатора Parabolic SAR

Классы управления капиталом и рисками	Описание
CMoneyFixedLot	Класс реализации алгоритма для входа в рынок фиксированным лотом, заданным при настройке
CMoneyFixedMargin	Класс реализации алгоритма для входа в рынок фиксированным уровнем маржи, заданным при настройке

© 2000-2025, MetaQuotes Ltd.

Классы управления капиталом и рисками	Описание
CMoneyFixedRisk	Класс реализации алгоритма для входа в рынок фиксированным уровнем риска, заданным при настройке
CMoneyNone	Класс реализации алгоритма для входа в рынок минимальным лотом
CMoneySizeOptimized	Класс реализации алгоритма для входа в рынок с объемом, определяемом результатами предыдущих сделок

© 2000-2025, MetaQuotes Ltd.

Базовые классы для создания экспертов

Этот раздел содержит технические детали работы с классами для создания и проверки торговых стратегий и описание соответствующих компонентов стандартной библиотеки MQL5.

Использование классов для создания и проверки торговых стратегий позволит сэкономить время при разработке (а особенно, проверке) торговых стратегий.

Стандартная библиотека MQL5 (в части набора классов для создания и проверки торговых стратегий) размещается в рабочем каталоге терминала в папке Include\Expert.

Класс	Описание
CExpertBase	Базовый класс для классов торговых стратегий
CExpert	Базовый класс для реализации торговых стратегий
CExpertSignal	Базовый класс для создания генераторов торговых сигналов
CExpertTrailing	Базовый класс для реализации алгоритмов сопровождения открытых позиций
CExpertMoney	Базовый класс для реализации алгоритмов управления капиталом и рисками

© 2000-2025, MetaQuotes Ltd.

Класс CExpertBase

Класс CExpertBase является базовым классом для класса CExpert и всех вспомогательных классов торговых стратегий.

Описание

Класс CExpertBase предоставляет набор данных и методов, общих для всех составных частей эксперта.

Декларация

```
class CExpertBase : public CObject
```

Заголовок

```
#include <Expert\ExpertBase.mqh>
```

Иерархия наследования

CObject

CExpertBase

Прямые потомки

CExpert, CExpertMoney, CExpertSignal, CExpertTrailing

Методы класса по группам

Публичные методы:

Инициализация	
virtual Init	Инициализирует объект
virtual ValidationSettings	Проверяет корректность настроек
Установка параметров	
Symbol	Устанавливает рабочий инструмент объекта
Period	Устанавливает рабочий таймфрейм объекта
Magic	Устанавливает идентификатор (magic) советника
Методы создания индикаторов и таймсерий	
virtual SetPriceSeries	Устанавливает указатели на внешние ценовые объекты-таймсерии.
virtual SetOtherSeries	Устанавливает указатели на внешние неценовые объекты-таймсерии
virtual InitIndicators	Инициализирует необходимые индикаторы и таймсерии

© 2000-2025, MetaQuotes Ltd.

Инициализация	
Доступ к защищенным данным	
InitPhase	Получает текущую фазу инициализации объекта
TrendType	Устанавливает идентификатор тренда
UsedSeries	Получает список используемых таймсерий
EveryTick	Устанавливает флаг анализа текущего бара
Доступ к таймсериям	
Open	Получает значение элемента таймсерии Open по указанному индексу
High	Получает значение элемента таймсерии High по указанному индексу
Low	Получает значение элемента таймсерии Low по указанному индексу
Close	Получает значение элемента таймсерии Close по указанному индексу
Spread	Получает значение элемента таймсерии Spread по указанному индексу
Time	Получает значение элемента таймсерии Time по указанному индексу
TickVolume	Получает значение элемента таймсерии TickVolume по указанному индексу
RealVolume	Получает значение элемента таймсерии RealVolume по указанному индексу

Защищенные методы:

Инициализация таймсерий	
InitOpen	Инициализирует таймсерию Open
InitHigh	Инициализирует таймсерию High
InitLow	Инициализирует таймсерию Low
InitClose	Инициализирует таймсерию Close
InitSpread	Инициализирует таймсерию Spread
InitTime	Инициализирует таймсерию Time
InitTickVolume	Инициализирует таймсерию TickVolume
InitRealVolume	Инициализирует таймсерию RealVolume

© 2000-2025, MetaQuotes Ltd.

Инициализация таймсерий	
Служебные методы	
virtual PriceLevelUnit	Получает единицу измерения ценовых уровней
virtual StartIndex	Получает номер индекса первого анализируемого бара
virtual CompareMagic	Сравнивает значение идентификатора (magic) с указанным значением

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

InitPhase

Получает текущую фазу инициализации объекта.

```
ENUM_INIT_PHASE  InitPhase()
```

Возвращаемое значение

Текущая фаза инициализации объекта.

Примечание

Инициализация объекта состоит из нескольких фаз (этапов). Назовем их условно:

1. Фаза начальной инициализации.

- начало - после завершения работы конструктора
- завершение - после удачного завершения работы метода Init(...)
- разрешено - вызов метода Init(...)
- запрещено - вызов любых других методов инициализации и метода ValidationSettings(...)

2. Фаза настройки параметров. Во время этой фазы необходимо установить все параметры объекта, которые будут использоваться для создания индикаторов.

- начало - после удачного завершения работы метода Init(...)
- завершение - после удачного завершения работы метода ValidationSettings(...)
- разрешено - вызовы методов Symbol(...) и Period(...)
- запрещено - вызовы методов Init(...), SetPriceSeries(...), SetOtherSeries(...) и InitIndicators(...)

3. Фаза проверки параметров.

- начало - после удачного завершения работы метода ValidationSettings()
- завершение - после удачного завершения работы метода InitIndicators(...)
- разрешено - вызовы методов Symbol(...), Period(...) и InitIndicators(...)
- запрещено - вызов любых других методов инициализации

4. Фаза завершения инициализации.

- начало - после удачного завершения работы метода InitIndicators(...)
- запрещено - вызов методов инициализации

© 2000-2025, MetaQuotes Ltd.

TrendType

Устанавливает идентификатор тренда.

```
void  TrendType(
   M_TYPE_TREND    value          // значение
   )
```

Параметры

value

[in] Новое значение идентификатора тренда.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

UsedSeries

Получает список используемых таймсерий.

```
int  UsedSeries()
```

Возвращаемое значение

Список используемых таймсерий в виде битовой карты.

Примечание

Если бит установлен, соответствующая таймсерия используется, если сброшен - нет.

Соответствие бит-таймсерия:

бит 0 - таймсерия Open,
бит 1 - таймсерия High,
бит 2 - таймсерия Low,
бит 3 - таймсерия Close,
бит 4 - таймсерия Spread,
бит 5 - таймсерия Time,
бит 6 - таймсерия TickVolume,
бит 7 - таймсерия RealVolume.

© 2000-2025, MetaQuotes Ltd.

EveryTick

Устанавливает флаг анализа текущего бара.

```
void  EveryTick(
   bool    value          // флаг
   )
```

Параметры

value

[in] Новое значение флага.

Возвращаемое значение

Нет.

Примечание

Если флаг установлен, обрабатывается каждое изменение цены (тик) по рабочему инструменту.

Если флаг сброшен, обработка производится только по факту формирования нового бара по рабочему инструменту на рабочем таймфрейме.

© 2000-2025, MetaQuotes Ltd.

Open

Получает значение элемента таймсерии Open по указанному индексу.

```
double  Open(
   int     ind          // индекс
   )
```

Параметры

ind

[in] Индекс элемента таймсерии.

Возвращаемое значение

Значение элемента таймсерии Open по указанному индексу случае успешного завершения, иначе EMPTY_VALUE.

Примечание

Значение EMPTY_VALUE может быть получено в двух случаях:

1. Таймсерия не используется (не был установлен соответствующий бит использования).
2. Индекс элемента вышел за пределы подготовленных данных таймсерии.

© 2000-2025, MetaQuotes Ltd.

High

Получает значение элемента таймсерии High по указанному индексу.

```
double  High(
   int     ind          // индекс
   )
```

Параметры

ind

[in] Индекс элемента таймсерии.

Возвращаемое значение

Значение элемента таймсерии High по указанному индексу случае успешного завершения, иначе EMPTY_VALUE.

Примечание

Значение EMPTY_VALUE может быть получено в двух случаях:

1. Таймсерия не используется (не был установлен соответствующий бит использования).
2. Индекс элемента вышел за пределы подготовленных данных таймсерии.

© 2000-2025, MetaQuotes Ltd.

Low

Получает значение элемента таймсерии Low по указанному индексу.

```
double  Low(
   int     ind          // индекс
   )
```

Параметры

ind

[in] Индекс элемента таймсерии.

Возвращаемое значение

Значение элемента таймсерии Low по указанному индексу случае успешного завершения, иначе EMPTY_VALUE.

Примечание

Значение EMPTY_VALUE может быть получено в двух случаях:

1. Таймсерия не используется (не был установлен соответствующий бит использования).
2. Индекс элемента вышел за пределы подготовленных данных таймсерии.

© 2000-2025, MetaQuotes Ltd.

Close

Получает значение элемента таймсерии Close по указанному индексу.

```
double  Close(
   int    ind          // индекс
   )
```

Параметры

ind

[in] Индекс элемента таймсерии.

Возвращаемое значение

Значение элемента таймсерии Close по указанному индексу случае успешного завершения, иначе EMPTY_VALUE.

Примечание

Значение EMPTY_VALUE может быть получено в двух случаях:

1. Таймсерия не используется (не был установлен соответствующий бит использования).
2. Индекс элемента вышел за пределы подготовленных данных таймсерии.

© 2000-2025, MetaQuotes Ltd.

Spread

Получает значение элемента таймсерии Spread по указанному индексу.

```
double  Spread(
   int     ind          // индекс
   )
```

Параметры

ind

[in] Индекс элемента таймсерии.

Возвращаемое значение

Значение элемента таймсерии Spread по указанному индексу случае успешного завершения, иначе EMPTY_VALUE.

Примечание

Значение EMPTY_VALUE может быть получено в двух случаях:

1. Таймсерия не используется (не был установлен соответствующий бит использования).
2. Индекс элемента вышел за пределы подготовленных данных таймсерии.

© 2000-2025, MetaQuotes Ltd.

Time

Получает значение элемента таймсерии Time по указанному индексу.

```
datetime  Time(
   int     ind          // индекс
   )
```

Параметры

ind

[in] Индекс элемента таймсерии.

Возвращаемое значение

Значение элемента таймсерии Time по указанному индексу случае успешного завершения, иначе EMPTY_VALUE.

Примечание

Значение EMPTY_VALUE может быть получено в двух случаях:

1. Таймсерия не используется (не был установлен соответствующий бит использования).
2. Индекс элемента вышел за пределы подготовленных данных таймсерии.

© 2000-2025, MetaQuotes Ltd.

TickVolume

Получает значение элемента таймсерии TickVolume по указанному индексу.

```
long  TickVolume(
   int     ind          // индекс
   )
```

Параметры

ind

[in] Индекс элемента таймсерии.

Возвращаемое значение

Значение элемента таймсерии TickVolume по указанному индексу случае успешного завершения, иначе EMPTY_VALUE.

Примечание

Значение EMPTY_VALUE может быть получено в двух случаях:

1. Таймсерия не используется (не был установлен соответствующий бит использования).
2. Индекс элемента вышел за пределы подготовленных данных таймсерии.

© 2000-2025, MetaQuotes Ltd.

RealVolume

Получает значение элемента таймсерии RealVolume по указанному индексу.

```
long  RealVolume(
   int    ind          // индекс
   )
```

Параметры

ind

[in] Индекс элемента таймсерии.

Возвращаемое значение

Значение элемента таймсерии RealVolume по указанному индексу случае успешного завершения, иначе EMPTY_VALUE.

Примечание

Значение EMPTY_VALUE может быть получено в двух случаях:

1. Таймсерия не используется (не был установлен соответствующий бит использования).
2. Индекс элемента вышел за пределы подготовленных данных таймсерии.

© 2000-2025, MetaQuotes Ltd.

Init

Инициализирует объект.

```
bool  Init(
   CSymbolInfo      symbol,    //
   ENUM_TIMEFRAMES  period,    //
   double           point      //
   )
```

Параметры

symbol

[in] Указатель на объект CSymbolInfo для доступа к информации о рабочем инструменте.

period

[in] Рабочий таймфрейм из перечисления ENUM_TIMEFRAMES.

point

[in] "Вес" 2/4 знакового пункта.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

© 2000-2025, MetaQuotes Ltd.

Symbol

Устанавливает рабочий инструмент объекта.

```
bool  Symbol(
   string    name      // символ
   )
```

Параметры

name

[in] Наименование рабочего инструмента.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Установка рабочего инструмента необходима в том случае если объект использует инструмент, отличный от установленного при начальной инициализации.

© 2000-2025, MetaQuotes Ltd.

Period

Устанавливает рабочий таймфрейм объекта.

```
bool  Period(
   ENUM_TIMEFRAMES  value     // таймфрейм
   )
```

Параметры

value

[in] Рабочий таймфрейм.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Установка рабочего таймфрейма необходима в том случае, если объект использует таймфрейм, отличный от установленного при начальной инициализации.

© 2000-2025, MetaQuotes Ltd.

Magic

Устанавливает идентификатор (magic) советника.

```
void  Magic(
   ulong  value     // идентификатор
   )
```

Параметры

value

[in] Идентификатор советника.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Проверяет корректность настроек.

```
virtual bool  ValidationSettings()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

SetPriceSeries

Устанавливает указатели на внешние ценовые объекты-таймсерии.

```
virtual bool  SetPriceSeries(
   CiOpen*    open,         // указатель
   CiHigh*    high,         // указатель
   CiLow*     low,          // указатель
   CiClose*   close         // указатель
   )
```

Параметры

open

[in] Указатель для доступа к таймсерии Open.

high

[in] Указатель для доступа к таймсерии High.

low

[in] Указатель для доступа к таймсерии Low.

close

[in] Указатель для доступа к таймсерии Close.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Установка указателей на объекты доступа к ценовым таймсериям необходима в том случае, если объект использует инструмент и таймфрейм (установленные при начальной инициализации) и ценовые таймсерии, необходимые для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

SetOtherSeries

Устанавливает указатели на внешние неценовые объекты-таймсерии.

```
virtual bool  SetOtherSeries(
   CiSpread*        spread,         // указатель
   CiTime*          time,           // указатель
   CiTickVolume*    tick_volume,    // указатель
   CiRealVolume*    real_volume     // указатель
   )
```

Параметры

spread

[in] Указатель для доступа к таймсерии Spread.

time

[in] Указатель для доступа к таймсерии Time.

tick_volume

[in] Указатель для доступа к таймсерии TickVolume.

real_volume

[in] Указатель для доступа к таймсерии RealVolume.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Установка указателей на объекты доступа к неценовым таймсериям необходима в том случае, если объект использует инструмент и таймфрейм (установленные при начальной инициализации) и неценовые таймсерии, необходимые для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Инициализирует необходимые индикаторы и таймсерии.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Необходимые таймсерии инициализируются в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации.

© 2000-2025, MetaQuotes Ltd.

InitOpen

Инициализирует таймсерию Open.

```
bool  InitOpen(
   CIndicators*  indicators    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Таймсерия Open инициализируется в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации и таймсерия необходима для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

InitHigh

Инициализирует таймсерию High.

```
bool  InitHigh(
   CIndicators*  indicators    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Таймсерия High инициализируется в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации и таймсерия необходима для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

InitLow

Инициализирует таймсерию Low.

```
bool  InitLow(
   CIndicators*  indicators    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Таймсерия Low инициализируется в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации и таймсерия необходима для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

InitClose

Инициализирует таймсерию Close.

```
bool  InitClose(
   CIndicators*  indicators    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Таймсерия Close инициализируется в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации и таймсерия необходима для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

InitSpread

Инициализирует таймсерию Spread.

```
bool  InitSpread(
   CIndicators*  indicators    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Таймсерия Spread инициализируется в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации и таймсерия необходима для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

InitTime

Инициализирует таймсерию Time.

```
bool  InitTime(
   CIndicators*  indicators    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Таймсерия Time инициализируется в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации и таймсерия необходима для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

InitTickVolume

Инициализирует таймсерию TickVolume.

```
bool  InitTickVolume(
   CIndicators*  indicators    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Таймсерия TickVolume инициализируется в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации и таймсерия необходима для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

InitRealVolume

Инициализирует таймсерию RealVolume.

```
bool  InitRealVolume(
   CIndicators*  indicators    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Таймсерия RealVolume инициализируется в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации и таймсерия необходима для дальнейшей работы.

© 2000-2025, MetaQuotes Ltd.

PriceLevelUnit

Получает единицу измерения ценовых уровней.

```
virtual double  PriceLevelUnit()
```

Возвращаемое значение

Единица измерения ценовых уровней.

Примечание

В базовом классе возвращается "вес" 2/4 знакового пункта.

© 2000-2025, MetaQuotes Ltd.

StartIndex

Получает номер индекса первого анализируемого бара.

```
virtual int  StartIndex()
```

Возвращаемое значение

Номер индекса первого анализируемого бара.

Примечание

Если флаг анализа текущего бара установлен, возвращается 0 (анализ будет производится с текущего бара). Если флаг анализа текущего бара сброшен, возвращается 1 (анализ будет производится с последнего сформировавшегося бара).

© 2000-2025, MetaQuotes Ltd.

CompareMagic

Сравнивает значение идентификатора (magic) с указанным значением.

```
virtual bool  CompareMagic(
   ulong  magic    // идентификатор
   )
```

Параметры

magic

[in] Значение идентификатора для сравнения.

Возвращаемое значение

true-в случае, если идентификатор совпадает с указанным, иначе false.

© 2000-2025, MetaQuotes Ltd.

Класс CExpert

Класс CExpert является базовым классом для реализации торговых стратегий.

В него уже заложены элементарные "навыки" торговли. Класс имеет встроенные механизмы для работы с таймсериями и индикаторами и набор виртуальных методов, определяющих торговую стратегию.

Для того чтобы ваш эксперт заработал по-другому, нужно:

1. Определиться с алгоритмами стратегии;

2. Создать свой класс, унаследовав его от CExpert;

3. Переопределить в своем классе виртуальные методы базового, заложив в них соответствующие алгоритмы.

Описание

Класс CExpert предоставляет набор виртуальных методов для реализации торговых стратегий.

Примечание

Позиция определяется принадлежащей эксперту и управляемой экспертом по паре свойств m_symbol и m_magic. В режиме "хеджинга" на счете одновременно может существовать несколько позиций по одному и тому же символу, поэтому особое внимание следует уделять значению m_magic.

Декларация

```
class CExpert : public CExpertBase
```

Заголовок

```
#include <Expert\Expert.mqh>
```

Иерархия наследования

CObject
  CExpertBase
    CExpert

Методы класса по группам

Инициализация	
Init	Инициализирует экземпляр класса
virtual InitSignal	Инициализирует объект торговых сигналов
virtual InitTrailing	Инициализирует объект трейлинга
virtual InitMoney	Инициализирует объект управления капиталом

© 2000-2025, MetaQuotes Ltd.

Инициализация	
virtual InitTrade	Инициализирует объект торговли
virtual ValidationSettings	Проверяет корректность настроек
virtual InitIndicators	Инициализирует необходимые индикаторы и таймсерии
virtual InitParameters	Инициализирует параметры эксперта
virtual Deinit	Деинициализирует экземпляр класса
virtual DeinitSignal	Деинициализирует объект торговых сигналов
virtual DeinitTrailing	Деинициализирует объект трейлинга
virtual DeinitMoney	Деинициализирует объект управления капиталом
virtual DeinitTrade	Деинициализирует объект торговли
virtual DeinitIndicators	Деинициализирует индикаторы и таймсерии
Установка параметров	
Magic	Устанавливает идентификатор эксперта
MaxOrders	Получает/Устанавливает максимально допустимое количество ордеров
OnTickProcess	Устанавливает флаг обработки события "OnTick"
OnTradeProcess	Устанавливает флаг обработки события "OnTrade"
OnTimerProcess	Устанавливает флаг обработки события "OnTimer"
OnChartEventProcess	Устанавливает флаг обработки события "OnChartEvent"
OnBookEventProcess	Устанавливает флаг обработки события "OnBookEvent"
Обработчики событий	
OnTick	Обработчик события OnTick
OnTrade	Обработчик события OnTrade
OnTimer	Обработчик события OnTimer
OnChartEvent	Обработчик события OnChartEvent
OnBookEvent	Обработчик события OnBookEvent
Обновление	
Refresh	Обновляет все необходимые данные
Основной метод обработки	
Processing	Реализует основной алгоритм
Методы входа в рынок	
CheckOpen	Проверяет необходимость и возможность входа в рынок

© 2000-2025, MetaQuotes Ltd.

Инициализация	
CheckOpenLong	Проверяет необходимость и возможность входа в длинную позицию
CheckOpenShort	Проверяет необходимость и возможность входа в короткую позицию
OpenLong	Выполняет операции для открытия длинной позиции
OpenShort	Выполняет операции для открытия короткой позиции
Методы выхода из рынка	
CheckClose	Проверяет необходимость выхода из рынка
CheckCloseLong	Проверяет необходимость выхода из длинной позиции
CheckCloseShort	Проверяет необходимость выхода из короткой позиции
CloseAll	Выходит из рынка полностью
Close	Выходит из рынка
CloseLong	Выходит из длинной позиции
CloseShort	Выходит из короткой позиции
Методы разворота позиции	
CheckReverse	Проверяет необходимость разворота позиции
CheckReverseLong	Проверяет необходимость разворота длинной позиции
CheckReverseShort	Проверяет необходимость разворота короткой позиции
ReverseLong	Производит разворот длинной позиции
ReverseShort	Производит разворот короткой позиции
Методы сопровождения позиций/ордеров	
CheckTrailingStop	Проверяет необходимость модификации параметров позиции
CheckTrailingStopLong	Проверяет необходимость модификации параметров длинной позиции
CheckTrailingStopShort	Проверяет необходимость модификации параметров короткой позиции
TrailingStopLong	Модифицирует параметры длинной позиции
TrailingStopShort	Модифицирует параметры короткой позиции
CheckTrailingOrderLong	Проверяет необходимость модификации параметров ордера на покупку

© 2000-2025, MetaQuotes Ltd.

Инициализация	
CheckTrailingOrderShort	Проверяет необходимость модификации параметров ордера на продажу
TrailingOrderLong	Модифицирует параметры ордера на покупку
TrailingOrderShort	Модифицирует параметры ордера на продажу
Методы удаления ордеров	
CheckDeleteOrderLong	Проверяет необходимость удаления ордеров на покупку
CheckDeleteOrderShort	Проверяет необходимость удаления ордеров на продажу
DeleteOrders	Удаляет все ордеры
DeleteOrder	Удаляет ордер
DeleteOrderLong	Удаляет ордер на покупку
DeleteOrderShort	Удаляет ордер на продажу
Методы определения объема	
LotOpenLong	Определяет объем торговой операции на покупку
LotOpenShort	Определяет объем торговой операции на продажу
LotReverse	Определяет объем торговой операции разворота позиции
Методы работы с торговой историей	
PrepareHistoryDate	Устанавливает начальную дату контролируемой истории торговли
HistoryPoint	Создает контрольную точку истории торговли
CheckTradeState	Обрабатывает изменение истории торговли
Методы работы с флагами торговых событий	
WaitEvent	Устанавливает флаг ожидания торгового события
NoWaitEvent	Сбрасывает флаг ожидания торгового события
Методы обработки торговых событий	
TradeEventPositionStopTake	Обработчик события "Срабатывание Stop Loss/Take Profit"
TradeEventOrderTriggered	Обработчик события "Отложенный ордер сработал"
TradeEventPositionOpened	Обработчик события "Открытие позиции"
TradeEventPositionVolumeChanged	Обработчик события "Добавление/уменьшение позиции"
TradeEventPositionModified	Обработчик события "Модификация параметров позиции"

© 2000-2025, MetaQuotes Ltd.

Инициализация	
TradeEventPositionClosed	Обработчик события "Закрытие позиции"
TradeEventOrderPlaced	Обработчик события "Отложенный ордер установлен"
TradeEventOrderModified	Обработчик события "Отложенный ордер модифицирован"
TradeEventOrderDeleted	Обработчик события "Отложенный ордер удален"
TradeEventNotIdentified	Обработчик неидентифицированного события
Служебные методы	
TimeframeAdd	Добавляет таймфрейм для контроля
TimeframesFlags	Формирует флаги таймфреймов
SelectPosition	Выбор позиции для последующей работы с ней.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

Init

Инициализирует экземпляр класса.

```
bool  Init(
   string              symbol,         // символ
   ENUM_TIMEFRAMES     period,         // период
   bool                every_tick,     // флаг
   ulong               magic           // идентификатор эксперта
   )
```

Параметры

symbol

[in] Рабочий символ эксперта.

period

[in] Рабочий период эксперта из перечисления ENUM_TIMEFRAMES.

every_tick

[in] Флаг.

magic

[in] Иденитификатор эксперта.

Возвращаемое значение

Нет.

Примечание

Если параметр every_tick установлен в true, то вызов метода Processing() производится на каждом тике рабочего символа. В противном случае вызов метода Processing() будет производиться только при формировании нового бара на рабочем периоде рабочего символа эксперта.

© 2000-2025, MetaQuotes Ltd.

Magic

Устанавливает идентификатор (magic) эксперта.

```
void  Magic(
   ulong  value     // новое значение
   )
```

Параметры

value

[in] Новое значение идентификатора эксперта.

Возвращаемое значение

Нет.

Примечание

При изменении идентификатора эксперта всем вспомогательным объектам присваивается то же значение идентификатора.

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

InitSignal

Инициализирует объект торговых сигналов.

```
virtual bool  InitSignal(
   CExpertSignal*     signal=NULL,        // указатель
   )
```

Параметры

signal

[in] Указатель на объект CExpertSignal или его потомка.

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Если параметр signal будет равен NULL, то объект торговых сигналов будет проинициализирован базовым классом CExpertSignal (который ничего не делает).

© 2000-2025, MetaQuotes Ltd.

InitTrailing

Инициализирует объект трейлинга.

```
virtual bool  InitTrailing(
   CExpertTrailing*     trailing=NULL,        // указатель
   )
```

Параметры

trailing

[in] Указатель на объект CExpertTrailing или его потомка.

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Если параметр trailing будет равен NULL, то объект трейлинга будет проинициализирован базовым классом ExpertTrailing (который ничего не делает).

© 2000-2025, MetaQuotes Ltd.

InitMoney

Инициализирует объект управления капиталом.

```
virtual bool  InitMoney(
   CExpertMoney*     money=NULL,         // указатель
   )
```

Параметры

money

[in] Указатель на объект CExpertMoney или его потомка.

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Если параметр money будет равен NULL, то объект управления капиталом будет проинициализирован базовым классом CExpertMoney (который всегда предлагает минимальный лот).

© 2000-2025, MetaQuotes Ltd.

InitTrade

Инициализирует объект торговли.

```
virtual bool  InitTrade(
   ulong             magic,        // идентификатор
   CExpertTrade*     trade=NULL    // указатель
   )
```

Параметры

magic

[in] Идентификатор эксперта, который будет использоваться при формировании торговых ордеров.

trade

[in] Указатель на объект торговли.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Init

Деинициализирует экземпляр класса.

```
virtual void Deinit()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnTickProcess

Устанавливает флаг обработки события OnTick.

```
void  OnTickOProcess(
   bool     value        // флаг
   )
```

Параметры

value

[in] Флаг обработки события OnTick.

Возвращаемое значение

Нет.

Примечание

При установке флага в состояние true, событие OnTick будет обрабатываться, в противном случае событие OnTick обрабатываться не будет. По умолчанию флаг установлен в состояние true.

© 2000-2025, MetaQuotes Ltd.

OnTradeProcess

Устанавливает флаг обработки события OnTrade.

```
void  OnTradeProcess(
   bool     value        // флаг
   )
```

Параметры

value

[in] Флаг обработки события OnTrade.

Возвращаемое значение

Нет.

Примечание

При установке флага в состояние true, событие OnTrade будет обрабатываться, в противном случае событие OnTrade обрабатываться не будет. По умолчанию флаг установлен в состояние false.

© 2000-2025, MetaQuotes Ltd.

OnTimerProcess

Устанавливает флаг обработки события OnTimer.

```
void  OnTimerProcess(
   bool     value        // флаг
   )
```

Параметры

value

[in] Флаг обработки события OnTimer.

Возвращаемое значение

Нет.

Примечание

При установке флага в состояние true, событие OnTimer будет обрабатываться, в противном случае событие OnTimer обрабатываться не будет. По умолчанию флаг установлен в состояние false.

© 2000-2025, MetaQuotes Ltd.

OnChartEventProcess

Устанавливает флаг обработки события OnChartEvent.

```
void  OnChartEventProcess(
   bool     value        // флаг
   )
```

Параметры

value

[in] Флаг обработки события OnChartEvent.

Возвращаемое значение

Нет.

Примечание

При установке флага в состояние true, событие OnChartEvent будет обрабатываться, в противном случае событие OnChartEvent обрабатываться не будет. По умолчанию флаг установлен в состояние false.

© 2000-2025, MetaQuotes Ltd.

OnBookEventProcess

Устанавливает флаг обработки события OnBookEvent.

```
void  OnChartEventProcess(
   bool     value        // флаг
   )
```

Параметры

value

[in] Флаг обработки события OnBookEvent.

Возвращаемое значение

Нет.

Примечание

При установке флага в состояние true, событие OnBookEvent будет обрабатываться, в противном случае событие OnBookEvent обрабатываться не будет. По умолчанию флаг установлен в состояние false.

© 2000-2025, MetaQuotes Ltd.

MaxOrders (Get Method)

Получает максимально допустимое количество ордеров.

```
int  MaxOrders()
```

Возвращаемое значение

Максимально допустимое количество ордеров.

MaxOrders (Set Method)

Устанавливает максимально допустимое количество ордеров.

```
void  MaxOrders(
   int      max_orders         // количество ордеров
   )
```

Параметры

max_orders

[in] Новое значение максимально допустимого количества ордеров.

Возвращаемое значение

Нет.

Примечание

По умолчанию максимально допустимое количество ордеров равно 1.

© 2000-2025, MetaQuotes Ltd.

Signal

Возвращает указатель на объект торговых сигналов.

```
CExpertSignal*  Signal() const
```

Возвращаемое значение

Указатель на объект торговых сигналов.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Проверяет корректность настроек.

```
virtual bool  ValidationSettings()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Проверяется корректность настроек всех вспомогательных объектов эксперта.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Инициализирует необходимые индикаторы и таймсерии.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Необходимые таймсерии инициализируются в том случае, если объект использует инструмент или таймфрейм, отличный от установленного при начальной инициализации.

Производится инициализация индикаторов и таймсерий всех вспомогательных объектов эксперта.

© 2000-2025, MetaQuotes Ltd.

OnTick

Обработчик события OnTick.

```
virtual void  OnTick()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnTrade

Обработчик события OnTrade.

```
virtual void  OnTrade()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnTimer

Обработчик события OnTimer.

```
virtual void  OnTimer()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnChartEvent

Обработчик события OnChartEvent.

```
virtual void  OnChartEvent(
   const int     id,       // идентификатор события
   const long&   lparam,   // параметр события типа long
   const double  dparam,   // параметр события типа double
   const string  sparam    // параметр события типа string
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Параметр события типа long.

dparam

[in] Параметр события типа double.

sparam

[in] Параметр события типа string.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnBookEvent

Обработчик события OnBookEvent.

```
virtual void  OnBookEvent(
   const string&     symbol        // символ
)
```

Параметры

symbol

[in] Символ, по которому пришло событие OnBookEvent.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

InitParameters

Инициализирует параметры эксперта.

```
virtual bool  InitParameters()
```

Возвращаемое значение

true-в случае удачи, иначе false.

Примечание

В классе CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

DeinitTrade

Деинициализирует объект торговли.

```
virtual void  DeinitTrade()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

DeinitSignal

Деинициализирует объект объект торговых сигналов.

```
virtual void  DeinitSignal()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

DeinitTrailing

Деинициализирует объект трейлинга.

```
virtual void  DeinitTrailing()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

DeinitMoney

Деинициализирует объект управления капиталом.

```
virtual void  DeinitMoney()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

DeinitIndicators

Деинициализирует индикаторы и таймсерии.

```
virtual void  DeinitIndicators()
```

Возвращаемое значение

Нет.

Примечание

Производится деинициализация индикаторов и таймсерий всех вспомогательных объектов эксперта.

© 2000-2025, MetaQuotes Ltd.

Refresh

Обновляет все необходимые данные.

```
virtual bool  Refresh()
```

Возвращаемое значение

true, если нужна дальнейшая обработка тика, иначе - false.

Примечание

Определяет необходимость и возможность обработки. Если обработка тика не нужна (или невозможна), возвращает false. Если обработка нужна, обновляет котировки и данные всех индикаторов и таймсерий и возвращает true.

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Processing

Реализует основной алгоритм.

```
virtual bool  Processing()
```

Возвращаемое значение

true - выполнена какая-либо торговая операция, иначе - false.

Примечание

Метод базового класса выполняет следующие действия:

1. Проверяется наличие открытой позиции по рабочему символу. Если позиции нет, то шаги 2, 3 и 4 пропускаются.

2. Проверяется необходимость переворота позиции (вызов метода CheckReverse()). Если позиция "перевернута", то уходим.

3. Проверяется необходимость закрытия позиции (вызов метода CheckClose()). Если позиция закрыта, то шаг 4 пропускается.

4. Проверяется необходимость модификации позиции (вызов метода CheckTrailingStop()). Если позиция модифицирована, то уходим.

5. Проверяется наличие отложенных ордеров по рабочему символу. Если ордеров нет, то переходим к п.9.

6. Проверяется необходимость удалить ордер (вызов метода CheckDeleteOrderLong() для ордеров на покупку или метода CheckDeleteOrderShort() для ордеров на продажу). Если ордер удален, то переходим к п.9.

7. Проверяется необходимость модифицировать ордер (вызов метода CheckTrailingOrderLong() для ордеров на покупку или метода CheckTrailingOrderShort() для ордеров на продажу). Если ордер модифицирован, то уходим.

8. Уходим.

9. Проверяется необходимость входа в рынок (вызов метода CheckOpen()).

Для реализации собственного алгоритма следует переопределить метод Processing() в классе-наследнике.

Реализация:

© 2000-2025, 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

SelectPosition

Выбор позиции для последующей работы с ней.

```
void  SelectPosition()
```

Возвращаемое значение

Отсутствует.

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpen

Проверяет необходимость и возможность входа в рынок.

```
virtual bool  CheckOpen()
```

Возвращаемое значение

true - выполнена какая-либо торговая операция, иначе - false.

Примечание

Последовательно проверяется необходимость входа в рынок в длинную позицию (вызов метода CheckOpenLong()), затем в короткую (вызов метода CheckOpenShort()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Проверяет необходимость и возможность входа в рынок в длинную позицию.

```
virtual bool  CheckOpenLong()
```

Возвращаемое значение

true - выполнена какая-либо торговая операция, иначе - false.

Примечание

Проверяется необходимость входа в рынок в длинную позицию (вызов метода CheckOpenLong() объекта торговых сигналов) и если условие входа выполнено, входит в рынок с параметрами, установленными объектом торговых сигналов (вызов метода OpenLong()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Проверяет необходимость и возможность входа в рынок в короткую позицию.

```
virtual bool  CheckOpenShort()
```

Возвращаемое значение

true - выполнена какая-либо торговая операция, иначе - false.

Примечание

Проверяется необходимость входа в рынок в короткую позицию (вызов метода CheckOpenShort() объекта торговых сигналов), и если условие входа выполнено, входит в рынок с параметрами, установленными объектом торговых сигналов (вызов метода OpenShort()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

OpenLong

Выполняет операции для входа в рынок в длинную позицию.

```
virtual bool  OpenLong(
   double    price,    // цена
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Параметры

price

[in] Цена входа в рынок.

sl

[in] Цена Stop Loss.

tp

[in] Цена Take Profit.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Размер лота для входа определяется вызовом метода LotOpenLong(). Если лот не равен 0.0, входит в рынок (вызов метода Buy(...) объекта торговли).

Реализация

```
//+------------------------------------------------------------------+
//| 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));
  }
```

© 2000-2025, MetaQuotes Ltd.

OpenShort

Выполняет операции для входа в рынок в короткую позицию.

```
virtual bool  OpenShort(
   double    price,    // цена
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Параметры

price

[in] Цена входа в рынок.

sl

[in] Цена Stop Loss.

tp

[in] Цена Take Profit.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Размер лота для входа определяется вызовом метода LotOpenShort(). Если лот не равен 0.0, входит в рынок (вызов метода Sell(...) объекта торговли).

Реализация

```
//+------------------------------------------------------------------+
//| 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));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverse

Проверяет необходимость и возможность разворота позиции.

```
virtual bool  CheckReverse()
```

Возвращаемое значение

true - выполнена какая-либо торговая операция, иначе - false.

Примечание

Последовательно проверяется необходимость разворота длинной позиции (вызов метода CheckReverseLong()), затем короткой (вызов метода CheckReverseShort()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverseLong

Проверяет необходимость и возможность разворота длинной позиции.

```
virtual bool  CheckReverseLong()
```

Возвращаемое значение

true - выполнена какая-либо торговая операция, иначе - false.

Примечание

Проверяется необходимость разворота длинной позиции (вызов метода CheckReverseLong() объекта торговых сигналов) и если условие разворота выполнено, осуществляет разворот длинной позиции с параметрами, установленными объектом торговых сигналов (вызов метода ReverseLong()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverseShort

Проверяет необходимость и возможность разворота короткой позиции.

```
virtual bool  CheckReverseLong()
```

Возвращаемое значение

true - выполнена какая-либо торговая операция, иначе - false.

Примечание

Проверяется необходимость разворота короткой позиции (вызов метода CheckReverseShort() объекта торговых сигналов) и если условие разворота выполнено, осуществляет разворот короткой позиции с параметрами, установленными объектом торговых сигналов (вызов метода ReverseShort()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

ReverseLong

Выполняет операции разворота длинной позиции.

```
virtual bool  ReverseLong(
   double    price,    // цена
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Параметры

price

[in] Цена входа в рынок.

sl

[in] Цена Stop Loss.

tp

[in] Цена Take Profit.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Размер лота для разворота длинной позиции определяется вызовом метода LotReverse(). Если лот не равен 0.0, производит разворот длинной позиции (вызов метода Sell(...) объекта торговли).

В "хеджинговом" режиме учета позиций разворот осуществляется закрытием существующей позиции и открытием новой позиции противоположного направления с остаточным объемом.

Реализация

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 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);
//---
   bool result=true;
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
     {
      //--- first close existing position
      lot-=m_position.Volume();
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
     }
   if(result)
      result=m_trade.Sell(lot,price,sl,tp);
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

ReverseShort

Выполняет операции разворота короткой позиции.

```
virtual bool  ReverseShort(
   double    price,    // цена
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Параметры

price

[in] Цена входа в рынок.

sl

[in] Цена Stop Loss.

tp

[in] Цена Take Profit.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Размер лота для разворота короткой позиции определяется вызовом метода LotReverse(). Если лот не равен 0.0, производит разворот короткой позиции (вызов метода Buy(...) объекта торговли).

В "хеджинговом" режиме учета позиций разворот осуществляется закрытием существующей позиции и открытием новой позиции противоположного направления с остаточным объемом.

Реализация

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 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);
//---
   bool result=true;
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
     {
      //--- first close existing position
      lot-=m_position.Volume();
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
     }
   if(result)
      result=m_trade.Buy(lot,price,sl,tp);
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckClose

Проверяет необходимость выхода из рынка.

```
virtual bool  CheckClose()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

1. Проверяет необходимость выхода из рынка по программному Stop Out (вызов метода CheckClose() объекта управления капиталом). Если условие выполняется, закрываем позицию, удаляем все ордера (вызов метода CloseAll() и уходим.
2. Проверяет необходимость закрытия длинной или короткой позиции (вызов метода CheckCloseLong() или CheckCloseShort() соответственно) и, если позиция закрыта, удаляет все ордера (вызов метода DeleteOrders()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckCloseLong

Проверяет необходимость выхода из длинной позиции.

```
virtual bool  CheckCloseLong()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Проверяет необходимость закрытия длинной позиции (вызов метода CheckCloseLong() объекта торговых сигналов) и, если условие выполнено, закрывает позицию (вызов метода CloseLong()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckCloseShort

Проверяет необходимость выхода из короткой позиции.

```
virtual bool  CheckCloseShort()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Проверяет необходимость закрытия короткой позиции (вызов метода CheckCloseShort() объекта торговых сигналов) и, если условие выполнено, закрывает позицию (вызов метода CloseShort()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseAll

Выходит из рынка полностью или частично.

```
virtual bool  CloseAll(
   double    lot    // лот
   )
```

Параметры

lot

[in] Размер лота, на который нужно сократить позицию.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

В "неттинговом" режиме учета позиций закрытие производится методами CExpertTrade::Buy или CExpertTrade::Sell. В "хеджинговом" режиме - методом CTrade::PositionClose, который также можно использовать и на счетах с неттинговой системой учета. Для удаления всех отложенных ордеров предназначен метод DeleteOrders().

Реализация

```
//+------------------------------------------------------------------+
//| 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=false;
//--- check for close operations
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
   else
     {
      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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Close

Выходит из рынка.

```
virtual bool  Close()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Закрывает позицию (вызов метода торгового объекта PositionClose()).

Реализация

```
//+------------------------------------------------------------------+
//| Position close                                                   |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::Close()
  {
   return(m_trade.PositionClose(m_symbol.Name()));
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseLong

Выходит из длинной позиции.

```
virtual bool  CloseLong(
   double    price    // цена
   )
```

Параметры

price

[in] Цена входа в рынок.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

В "неттинговом" режиме учета позиций закрытие производится методами CExpertTrade::Buy или CExpertTrade::Sell. В "хеджинговом" режиме - методом CTrade::PositionCloseByTicket.

Реализация

```
//+------------------------------------------------------------------+
//| Long position close                                              |
//| INPUT:  price - price for close.                                 |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CloseLong(double price)
  {
   bool result=false;
//---
   if(price==EMPTY_VALUE)
      return(false);
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
   else
      result=m_trade.Sell(m_position.Volume(),price,0,0);
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseShort

Выходит из короткой позиции.

```
virtual bool  CloseShort(
   double    price    // цена
   )
```

Параметры

price

[in] Цена входа в рынок.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

В "неттинговом" режиме учета позиций закрытие производится методами CExpertTrade::Buy или CExpertTrade::Sell. В "хеджинговом" режиме - методом CTrade::PositionCloseByTicket.

Реализация

```
//+------------------------------------------------------------------+
//| Short position close                                             |
//| INPUT:  price - price for close.                                 |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CloseShort(double price)
  {
   bool result=false;
//---
   if(price==EMPTY_VALUE)
      return(false);
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
   else
      result=m_trade.Buy(m_position.Volume(),price,0,0);
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStop

Проверяет необходимость модификации параметров позиции.

```
virtual bool  CheckTrailingStop()
```

Возвращаемое значение

true - выполнена какая-либо торговая операция, иначе - false.

Примечание

Проверяется необходимость модификации позиции (вызов метода CheckTrailingStopLong() для длинной позиции или CheckTrailingStopShort() для короткой позиции).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Проверяет необходимость модификации параметров длинной позиции.

```
virtual bool  CheckTrailingStopLong()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Проверяется необходимость модификации длинной позиции (вызов метода CheckTrailingStopLong() объекта трейлинга). В случае выполнения условия модифицирует параметры длинной позиции в соответствии с установками объекта трейлинга (вызов метода TrailingStopLong()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Проверяет необходимость модификации параметров короткой позиции.

```
virtual bool  CheckTrailingStopShort()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Проверяется необходимость модификации коротокой позиции (вызов метода CheckTrailingStopShort() объекта трейлинга). В случае выполнения условия модифицирует параметры короткой позиции в соответствии с установками объекта трейлинга (вызов метода TrailingStopShort()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingStopLong

Модифицирует параметры длинной позиции.

```
virtual bool  TrailingStopLong(
   double    sl,    // цена Stop Loss
   double    tp,    // цена Take Profit
   )
```

Параметры

sl

[in] Цена Stop Loss.

tp

[in] Цена Take Profit.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Модифицирует позицию (вызов метода PositionModify() объекта торговли).

Реализация

```
//+------------------------------------------------------------------+
//| 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));
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingStopShort

Модифицирует параметры короткой позиции.

```
virtual bool  TrailingStopLong(
   double    sl,    // цена Stop Loss
   double    tp,    // цена Take Profit
   )
```

Параметры

sl

[in] Цена Stop Loss.

tp

[in] Цена Take Profit.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Модифицирует позицию (вызов метода PositionModify() объекта торговли).

Реализация

```
//+------------------------------------------------------------------+
//| 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));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderLong

Проверяет необходимость модификации параметров ордера на покупку.

```
virtual bool  CheckTrailingOrderLong()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Проверяется необходимость модификации параметров ордера на покупку (вызов метода CheckTrailingOrderLong() объекта торговых сигналов). В случае выполнения условия модифицирует параметры ордера (вызов метода TrailingOrderLong()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderShort

Проверяет необходимость модификации параметров ордера на продажу.

```
virtual bool  CheckTrailingOrderShort()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Проверяется необходимость модификации параметров ордера на продажу (вызов метода CheckTrailingOrderShort() объекта торговых сигналов). В случае выполнения условия модифицирует параметры ордера (вызов метода TrailingOrderShort()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingOrderLong

Модифицирует параметры ордера на покупку.

```
virtual bool  TrailingOrderLong(
   double    delta    // смещение
   )
```

Параметры

delta

[in] Изменение цены.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Модифицирует параметры ордера (вызов метода OrderModify(...) объекта торговли).

Реализация

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

© 2000-2025, MetaQuotes Ltd.

TrailingOrderShort

Модифицирует параметры ордера на продажу.

```
virtual bool  TrailingOrderShort(
   double    delta    // смещение
   )
```

Параметры

delta

[in] Изменение цены.

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Модифицирует параметры ордера (вызов метода OrderModify(...) объекта торговли).

Реализация

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

© 2000-2025, MetaQuotes Ltd.

CheckDeleteOrderLong

Проверяет необходимость удаления ордера на покупку.

```
virtual bool  CheckDeleteOrderLong()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

1. Проверяет время истечения ордера.
2. Проверяет необходимость удаления ордера (вызов метода CheckCloseLong() объекта торговых сигналов). В случае выполнения одного из условий, удаляет ордер (вызов метода DeleteOrderLong()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckDeleteOrderShort

Проверяет необходимость удаления ордера на продажу.

```
virtual bool  CheckDeleteOrderShort()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

1. Проверяет время истечения ордера.
2. Проверяет необходимость удаления ордера (вызов метода CheckCloseShort() объекта торговых сигналов). В случае выполнения одного из условий, удаляет ордер (вызов метода DeleteOrderShort()).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrders

Удаляет все ордеры.

```
virtual bool  DeleteOrders()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Удаляет все ордеры (вызов в цикле метода DeleteOrder).

Реализация

```
//+------------------------------------------------------------------+
//| 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrder

Удаляет ордер.

```
virtual bool  DeleteOrder()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Удаляет ордер (вызов метода OrderDelete(...) объекта торговли).

Реализация

```
//+------------------------------------------------------------------+
//| 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()));
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrderLong

Удаляет ордер на покупку.

```
virtual bool  DeleteOrderLong()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Удаляет ордер на покупку (вызов метода OrderDelete(..) объекта торговли).

Реализация

```
//+------------------------------------------------------------------+
//| 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()));
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrderShort

Удаляет ордер на продажу.

```
virtual bool  DeleteOrderShort()
```

Возвращаемое значение

true - выполнена торговая операция, иначе - false.

Примечание

Удаляет ордер на продажу (вызов метода OrderDelete(...) объекта торговли).

Реализация

```
//+------------------------------------------------------------------+
//| 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()));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotOpenLong

Определяет объем торговой операции на покупку.

```
double  LotOpenLong(
   double    price,    // цена
   double    sl        // цена Stop Loss
   )
```

Параметры

price

[in] Цена входа в рынок.

sl

[in] Цена Stop Loss.

Возвращаемое значение

Объем (в лотах) для операции на покупку.

Примечание

Определяет объем торговой операции на покупку (вызов метода CheckOpenLong(...) объекта управления капиталом).

Реализация

```
//+------------------------------------------------------------------+
//| 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));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotOpenShort

Определяет объем торговой операции на продажу.

```
double  LotOpenShort(
   double    price,    // цена
   double    sl        // цена Stop Loss
   )
```

Параметры

price

[in] Цена входа в рынок.

sl

[in] Цена Stop Loss.

Возвращаемое значение

Объем (в лотах) для операции на продажу.

Примечание

Определяет объем торговой операции на продажу (вызов метода CheckOpenShort(...) объекта управления капиталом).

Реализация

```
//+------------------------------------------------------------------+
//| 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));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotReverse

Определяет объем торговой операции разворота позиции.

```
double  LotReverse(
   double    sl       // цена Stop Loss
   )
```

Параметры

sl

[in] Цена Stop Loss.

Возвращаемое значение

Объем (в лотах) для операции разворота позиции.

Примечание

Определяет объем торговой операции разворота позиции (вызов метода CheckReverse(...) объекта управления капиталом).

Реализация

```
//+------------------------------------------------------------------+
//| 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));
  }
```

© 2000-2025, MetaQuotes Ltd.

PrepareHistoryDate

Устанавливает начальную дату контролируемой истории торговли.

```
void  PrepareHistoryDate()
```

Примечание

По умолчанию дата начала контролируемой истории устанавливается с начала месяца (но не менее одних суток).

© 2000-2025, MetaQuotes Ltd.

HistoryPoint

Создает контрольную точку истории торговли.

```
void  HistoryPoint(
   bool   from_check_trade=false   // флаг
   )
```

Параметры

from_check_trade=false

[in] Флаг для предотвращения рекурсии.

Примечание

Запоминает количество позиций, ордеров, сделок и исторических ордеров.

© 2000-2025, MetaQuotes Ltd.

CheckTradeState

Обрабатывает изменение истории торговли.

```
bool  CheckTradeState()
```

Возвращаемое значение

true - если событие обработанно, иначе - false.

Примечание

Проверяет количество позиций, ордеров, сделок и исторических ордеров, сравнивая их с ранее сохраненными в методе HistoryPoint(). В случае изменения вызывает соответствующий виртуальный обработчик.

© 2000-2025, MetaQuotes Ltd.

WaitEvent

Устанавливает флаг ожидания торгового события.

```
void  WaitEvent(
   ENUM_TRADE_EVENTS     event          // флаг
   )
```

Параметры

event

[in] Флаг ожидания торгового события (из перечисления ENUM_TRADE_EVENTS), который необходимо установить.

Возвращаемое значение

Нет.

Флаги событий

```
//--- 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 "triggeri
   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 "triggeri
  };
```

© 2000-2025, MetaQuotes Ltd.

NoWaitEvent

Сбрасывает флаг ожидания торгового события.

```
void  NoWaitEvent(
   ENUM_TRADE_EVENTS     event          // флаг
   )
```

Параметры

event

[in] Флаг ожидания торгового события (из перечисления ENUM_TRADE_EVENTS), который нужно сбросить.

Возвращаемое значение

Нет.

Флаги событий

```
//--- 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 "triggeri
   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 "triggeri
  };
```

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionStopTake

Обработчик события "Срабатывание Stop Loss/Take Profit".

```
virtual bool  TradeEventPositionStopTake()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderTriggered

Обработчик события "Отложенный ордер сработал".

```
virtual bool  TradeEventOrderTriggered()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionOpened

Обработчик события "Открытие позиции".

```
virtual bool  TradeEventPositionOpened()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionVolumeChanged

Обработчик события "Добавление/уменьшение позиции".

```
virtual bool  TradeEventPositionVolumeChanged()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionModified

Обработчик события "Модификация параметров позиции".

```
virtual bool  TradeEventPositionModified()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionClosed

Обработчик события "Закрытие позиции".

```
virtual bool  TradeEventPositionClosed()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderPlaced

Обработчик события "Отложенный ордер установлен".

```
virtual bool  TradeEventOrderPlaced()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderModified

Обработчик события "Отложенный ордер модифицирован".

```
virtual bool  TradeEventOrderModified()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderDeleted

Обработчик события "Отложенный ордер удален".

```
virtual bool  TradeEventOrderDeleted()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

TradeEventNotIdentified

Обработчик неидентифицированного события.

```
virtual bool  TradeEventNotIdentified()
```

Возвращаемое значение

Метод класса CExpert ничего не делает и всегда возвращает true.

Примечание

Следует отметить, что торговые события далеко не всегда бывают одиночными, чаще всего события от сервера поступают "пачками". В этом случае однозначная идентификация возникших торговых событий затруднена.

© 2000-2025, MetaQuotes Ltd.

TimeframeAdd

Добавляет таймфрейм для контроля.

```
void  TimeframeAdd(
   ENUM_TIMEFRAMES    period          // период
   )
```

Параметры

period

[in] Период (из перечисления ENUM_TIMEFRAMES), который необходимо контролировать.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

TimeframesFlags

Формирует флаги таймфреймов.

```
int  TimeframesFlags(
   MqlDateTime&     time          // ссылка
   )
```

Параметры

time

[in] Ссылка на структуру типа MqlDateTime, в которой содержится новое время.

Возвращаемое значение

Флаги таймфреймов, для которых пришел "новый бар".

© 2000-2025, MetaQuotes Ltd.

Класс CExpertSignal

Класс CExpertSignal является базовым классом для создания генераторов торговых сигналов, поэтому он, предоставляя интерфейсы, сам ничего не делает (исключение составляют методы CheckReverseLong() и CheckReverseShort()).

Для того чтобы генератор торговых сигналов "загенерировал", нужно:

1. Определиться с алгоритмами генерации торговых сигналов;
2. Создать свой класс генератора, унаследовав его от CExpertSignal;
3. Переопределить в своем классе виртуальные методы базового, заложив в них соответствующие алгоритмы.

В качестве примера можно рассмотреть любой mqh-файл из папки Expert\Signal\.

Описание

Класс CExpertSignal является основой для реализации алгоритмов генерации торговых сигналов.

Декларация

```
class CExpertSignal : public CExpertBase
```

Заголовок

```
#include <Expert\ExpertSignal.mqh>
```

Иерархия наследования

CObject

CExpertBase

CExpertSignal

Прямые потомки

CSignalAC, CSignalAMA, CSignalAO, CSignalBearsPower, CSignalBullsPower, CSignalCCI, CSignalDeM, CSignalDEMA, CSignalEnvelopes, CSignalFrAMA, CSignalRSI, CSignalRVI, CSignalSAR, CSignalStoch, CSignalTEMA, CSignalTriX, CSignalWPR

Методы класса по группам

Инициализация	
virtual InitIndicators	Инициализирует необходимые индикаторы и таймсерии
virtual ValidationSettings	Проверяет корректность настроек объекта
virtual AddFilter	Добавляет фильтр в комбинированный сигнал
Доступ к защищенным данным	

© 2000-2025, MetaQuotes Ltd.

Инициализация	
BasePrice	Устанавливает базовый уровень цены
UsedSeries	Получает флаги используемых таймсерий
Установка параметров	
Weight	Устанавливает значение параметра "Weight"
PatternsUsage	Устанавливает значение параметра "PatternsUsage"
General	Устанавливает значение параметра "General"
Ignore	Устанавливает значение параметра "Ignore"
Invert	Устанавливает значение параметра "Invert"
ThresholdOpen	Устанавливает значение параметра "ThresholdOpen"
ThresholdClose	Устанавливает значение параметра "ThresholdClose"
PriceLevel	Устанавливает значение параметра "PriceLevel"
StopLevel	Устанавливает значение параметра "StopLevel"
TakeLevel	Устанавливает значение параметра "TakeLevel"
Expiration	Устанавливает значение параметра "Expiration"
Magic	Устанавливает значение параметра "Magic"
Методы проверки необходимости открытия/разворота/закрытия позиций	
virtual CheckOpenLong	Определяет необходимость открытия длинной позиции
virtual CheckCloseLong	Определяет необходимость закрытия длинной позиции
virtual CheckOpenShort	Определяет необходимость открытия короткой позиции
virtual CheckCloseShort	Определяет необходимость закрытия короткой позиции
virtual CheckReverseLong	Определяет необходимость разворота длинной позиции
virtual CheckReverseShort	Определяет необходимость разворота короткой позиции
Методы установки параметров	
virtual OpenLongParams	Устанавливает параметры открытия длинной позиции
virtual OpenShortParams	Устанавливает параметры открытия короткой позиции
virtual CloseLongParams	Устанавливает параметры закрытия длинной позиции
virtual CloseShortParams	Устанавливает параметры закрытия короткой позиции

© 2000-2025, MetaQuotes Ltd.

Инициализация	
Методы проверки необходимости управления отложенными ордерами	
virtual CheckTrailingOrderLong	Определяет необходимость модификации параметров отложенного ордера на покупку
virtual CheckTrailingOrderShort	Определяет необходимость модификации параметров отложенного ордера на продажу
Методы проверки формирования рыночных моделей	
virtual LongCondition	Вовращает результат проверки возникновения условия на покупку
virtual ShortCondition	Вовращает результат проверки возникновения условия на продажу
virtual Direction	Возвращает значение "взвешенного" направления движения цены

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

BasePrice

Устанавливает базовый уровень цены.

```
void  BasePrice(
   double    value          // значение
   )
```

Параметры

value

[in] Новое значение параметра.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

UsedSeries

Получает флаги используемых таймсерий.

```
int  UsedSeries()
```

Возвращаемое значение

Флаги используемых таймсерий (если не установлен другой символ или другой таймфрейм), иначе 0.

© 2000-2025, MetaQuotes Ltd.

Weight

Устанавливает значение параметра "Weight".

```
void  Weight(
   double    value          // значение
   )
```

Параметры

value

[in] Параметр "Weight".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

PatternUsage

Устанавливает значение параметра "PatternsUsage".

```
void  PatternUsage(
   double    value         // значение
   )
```

Параметры

value

[in] Новое значение параметра "PatternsUsage".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

General

Устанавливает значение параметра "General".

```
void  General(
   int     value          // значение
   )
```

Параметры

value

[in] Новое значение параметра "General".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Ignore

Устанавливает значение параметра "Ignore".

```
void  Ignore(
   long     value          // значение
   )
```

Параметры

value

[in] Новое значение параметра "Ignore".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Invert

Устанавливает значение параметра "Invert".

```
void  Invert(
   long     value          // значение
   )
```

Параметры

value

[in] Новое значение параметра "Invert".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

ThresholdOpen

Устанавливает значение параметра "ThresholdOpen".

```
void  ThresholdOpen(
   long     value          // значение
   )
```

Параметры

value

[in] Новое значение параметра "ThresholdOpen".

Возвращаемое значение

Нет.

Примечание

Параметр "ThresholdOpen" может принимать значения от 0 до 100. Используется для определения необходимости открытия позиции по результатам "голосования".

© 2000-2025, MetaQuotes Ltd.

ThresholdClose

Устанавливает значение параметра "ThresholdClose".

```
void  ThresholdOpen(
   long     value          // значение
   )
```

Параметры

value

[in] Новое значение параметра "ThresholdClose".

Возвращаемое значение

Нет.

Примечание

Параметр "ThresholdClose" может принимать значения от 0 до 100. Используется для определения необходимости закрытия позиции по результатам "голосования".

© 2000-2025, MetaQuotes Ltd.

PriceLevel

Устанавливает значение параметра "PriceLevel".

```
void  PriceLevel(
   double    value         // значение
   )
```

Параметры

value

[in] Новое значение параметра "PriceLevel".

Возвращаемое значение

Нет.

Примечание

Параметр "PriceLevel" задается в единицах измерения ценовых уровней. Значение единицы измерения ценовых уровней возвращает метод PriceLevelUnit(). Используется для задания уровня открытия относительно базовой цены.

© 2000-2025, MetaQuotes Ltd.

StopLevel

Устанавливает значение параметра "StopLevel".

```
void  StopLevel(
   double    value          // значение
   )
```

Параметры

value

[in] Новое значение параметра "StopLevel".

Возвращаемое значение

Нет.

Примечание

Параметр "StopLevel" задается в единицах измерения ценовых уровней. Значение единицы измерения ценовых уровней возвращает метод PriceLevelUnit(). Используется для задания уровня фиксации убытков относительно цены открытия.

© 2000-2025, MetaQuotes Ltd.

TakeLevel

Устанавливает значение параметра "TakeLevel".

```
void  TakeLevel(
   double    value         // значение
   )
```

Параметры

value

[in] Новое значение параметра "TakeLevel".

Возвращаемое значение

Нет.

Примечание

Параметр "TakeLevel" задается в единицах измерения ценовых уровней. Значение единицы измерения ценовых уровней возвращает метод PriceLevelUnit(). Используется для задания уровня фиксации прибыли относительно цены открытия.

© 2000-2025, MetaQuotes Ltd.

Expiration

Устанавливает значение параметра "Expiration".

```
void  Expiration(
   int     value          // значение
   )
```

Параметры

value

[in] Новое значение параметра "Expiration".

Возвращаемое значение

Нет.

Примечание

Параметр "Expiration" задается в барах. Используется при расчёте времени истечения отложенного ордера (если предполагается вход не по текущей рыночной цене).

© 2000-2025, MetaQuotes Ltd.

Magic

Устанавливает значение параметра "Magic".

```
void  Magic(
   int    value         // значение
   )
```

Параметры

value

[in] Новое значение параметра "Magic" (идентификатор эксперта).

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Проверяет корректность настроек объекта.

```
virtual bool  ValidationSettings()
```

Возвращаемое значение

true - если настройки объекта корректны, иначе - false.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Инициализирует необходимые индикаторы и таймсерии.

```
virtual bool  InitIndicators(
   CIndicators*  indicators    // указатель
   )
```

Параметры

indicators

[in] Указатель на объект-коллекцию индикаторов и таймсерий.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

Примечание

Необходимые таймсерии инициализируются в том случае, если объект использует инструмент или таймфрейм,отличный от установленного при начальной инициализации.

© 2000-2025, MetaQuotes Ltd.

AddFilter

Добавляет фильтр в комбинированный сигнал.

```
virtual bool  AddFilter(
   CExpertSignal*  filter    // указатель
   )
```

Параметры

filter

[in] Указатель на объект-фильтр.

Возвращаемое значение

true-в случае успешного завершения, иначе false.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Определяет необходимость открытия длинной позиции.

```
virtual bool  CheckOpenLong(
   double&    price,          // ссылка
   double&    sl,             // ссылка
   double&    tp,             // ссылка
   datetime&  expiration      // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены открытия.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

expiration

[in][out] Ссылка на переменную для размещения времени истечения ордера (в случае необходимости).

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Определяет необходимость открытия короткой позиции.

```
virtual bool  CheckOpenShort(
   double&    price,          // ссылка
   double&    sl,             // ссылка
   double&    tp,             // ссылка
   datetime&  expiration      // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены открытия.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

expiration

[in][out] Ссылка на переменную для размещения времени истечения ордера (в случае необходимости).

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OpenLongParams

Устанавливает параметры открытия длинной позиции.

```
virtual bool  OpenLongParams(
   double&    price,          // ссылка
   double&    sl,             // ссылка
   double&    tp,             // ссылка
   datetime&  expiration      // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены открытия.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

expiration

[in][out] Ссылка на переменную для размещения времени истечения ордера (в случае необходимости).

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OpenShortParams

Устанавливает параметры открытия короткой позиции.

```
virtual bool  OpenShortParams(
   double&    price,          // ссылка
   double&    sl,             // ссылка
   double&    tp,             // ссылка
   datetime&  expiration      // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены открытия.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

expiration

[in][out] Ссылка на переменную для размещения времени истечения ордера (в случае необходимости).

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CheckCloseLong

Определяет необходимость закрытия длинной позиции.

```
virtual bool  CheckCloseLong(
   double&  price      // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены закрытия.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CheckCloseShort

Определяет необходимость закрытия короткой позиции.

```
virtual bool  CheckCloseShort(
   double&  price      // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены закрытия.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CloseLongParams

Устанавливает параметры закрытия длинной позиции.

```
virtual bool  CloseLongParams(
   double&    price          // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены закрытия.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CloseShortParams

Устанавливает параметры закрытия короткой позиции.

```
virtual bool  CloseShortParams(
   double&    price          // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены закрытия.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CheckReverseLong

Определяет необходимость разворота длинной позиции.

```
virtual bool  CheckReverseLong(
   double&    price,          // ссылка
   double&    sl,             // ссылка
   double&    tp,             // ссылка
   datetime&  expiration      // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены разворота.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

expiration

[in][out] Ссылка на переменную для размещения времени истечения ордера (в случае необходимости).

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CheckReverseShort

Определяет необходимость разворота короткой позиции.

```
virtual bool  CheckReverseShort(
   double&    price,          // ссылка
   double&    sl,             // ссылка
   double&    tp,             // ссылка
   datetime&  expiration      // ссылка
   )
```

Параметры

price

[in][out] Ссылка на переменную для размещения цены разворота.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

expiration

[in][out] Ссылка на переменную для размещения времени истечения ордера (в случае необходимости).

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderLong

Определяет необходимость модификации параметров отложенного ордера на покупку.

```
virtual bool  CheckTrailingOrderLong(
   COrderInfo*     order,          // указатель
   double&         price           // ссылка
   )
```

Параметры

order

[in] Указатель на объект COrderInfo.

price

[in][out] Ссылка на переменную для размещения цены Stop Loss.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderShort

Определяет необходимость модификации параметров отложенного ордера на продажу.

```
virtual bool  CheckTrailingOrderShort(
   COrderInfo*     order,          // указатель
   double&         price           // ссылка
   )
```

Параметры

order

[in] Указатель на объект COrderInfo.

price

[in][out] Ссылка на переменную для размещения цены Stop Loss.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

© 2000-2025, MetaQuotes Ltd.

LongCondition

Вовращает результат проверки возникновения условия на покупку.

```
virtual int  LongCondition()
```

Возвращаемое значение

В случае возникновения условия на покупку, возвращает число от 1 до 100 (чем больше значение, тем больше "сила" сигнала), в случае отсутствия сигнала, возваращается 0.

Примечание

Базовый класс не имеет реализацию алгоритма определения условия на покупку, поэтому метод базового класса всегда возвращает 0.

© 2000-2025, MetaQuotes Ltd.

ShortCondition

Вовращает результат проверки возникновения условия на продажу.

```
virtual int  ShortCondition()
```

Возвращаемое значение

В случае возникновения условия на продажу, возвращает число от 1 до 100 (чем больше значение, тем больше "сила" сигнала), в случае отсутствия сигнала, возваращается 0.

Примечание

Базовый класс не имеет реализацию алгоритма определения условия на продажу, поэтому метод базового класса всегда возвращает 0.

© 2000-2025, MetaQuotes Ltd.

Direction

Возвращает значение "взвешенного" направления движения цены.

```
virtual double  Direction()
```

Возвращаемое значение

В случае вероятного движения цены вверх, значение будет >0, в случае вероятного движения цены вниз, значение будет <0. Чем больше абсолютное значение, тем "сильнее" сигнал.

Примечание

При наличии встроенных фильтров, их результат будет учтен при определении общего направления.

© 2000-2025, MetaQuotes Ltd.

Класс CExpertTrailing

Класс CExpertTrailing является базовым классом для реализации алгоритмов сопровождения открытых позиций, поэтому он, предоставляя интерфейсы, сам ничего не делает.

Для того чтобы "трейлинг" заработал по-другому, нужно:

1. Определиться с алгоритмами сопровождения открытых позиций;
2. Создать свой класс, унаследовав его от CExpertTrailing;
3. Переопределить в своем классе виртуальные методы базового, заложив в них соответствующие алгоритмы.

В качестве примера можно рассмотреть любой mqh-файл из папки Expert\Trailing\.

Описание

Класс CExpertTrailing является основой для реализации алгоритмов сопровождения открытых позиций.

Декларация

```
class CExpertTrailing : public CExpertBase
```

Заголовок

```
#include <Expert\ExpertTrailing.mqh>
```

Иерархия наследования

CObject

CExpertBase

CExpertTrailing

Прямые потомки

CTrailingFixedPips, CTrailingMA, CTrailingNone, CTrailingPSAR

Методы класса по группам

Методы проверки необходимости управления отложенными ордерами	
virtual CheckTrailingStopLong	Определяет необходимость модификации параметров длинной позиции
virtual CheckTrailingStopShort	Определяет необходимость модификации параметров короткой позиции

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Определяет необходимость модификации параметров длинной позиции.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // указатель
   double&         sl,            // ссылка для Stop Loss
   double&         tp             // ссылка для Take Profit
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Метод базового класса всегда возвращает false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Определяет необходимость модификации параметров короткой позиции.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // указатель
   double&         sl,           // ссылка для Stop Loss
   double&         tp            // ссылка для Take Profit
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Метод базового класса всегда возвращает false.

© 2000-2025, MetaQuotes Ltd.

Класс CExpertMoney

Класс CExpertMoney является базовым классом для реализации алгоритмов управления капиталом и рисками.

Описание

Класс CExpertMoney является основой для реализации алгоритмов управления капиталом и рисками.

Декларация

```
class CExpertMoney : public CObject
```

Заголовок

```
#include <Expert\ExpertMoney.mqh>
```

Иерархия наследования

CObject

CExpertBase

CExpertMoney

Прямые потомки

CMoneyFixedLot, CMoneyFixedMargin, CMoneyFixedRisk, CMoneyNone, CMoneySizeOptimized

Методы класса по группам

Доступ к защищенным данным	
Percent	Устанавливает значение параметра "Процент риска"
Инициализация	
virtual ValidationSettings	Проверяет корректность настроек
Методы проверки необходимости открытия/разворота/закрытия позиций	
virtual CheckOpenLong	Определяет объем для открытия длинной позиции
virtual CheckOpenShort	Определяет объем для открытия короткой позиции
virtual CheckReverse	Определяет объем для разворота позиции
virtual CheckClose	Определяет необходимость закрытия позиции

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2025, MetaQuotes Ltd.

Percent

Устанавливает процент риска.

```
void  Percent(
   double  percent      // процент риска
   )
```

Параметры

percent

[in] Процент риска.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Проверяет корректность настроек.

```
virtual bool  ValidationSettings()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Метод базового класса всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Определяет объем для открытия длинной позиции.

```
virtual double  CheckOpenLong(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Цена открытия длинной позиции.

sl

[in] Цена Stop Loss длинной позиции.

Возвращаемое значение

Объем для открытия длинной позиции.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Определяет объем для открытия короткой позиции.

```
virtual double  CheckOpenShort(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Цена открытия короткой позиции.

sl

[in] Цена Stop Loss короткой позиции.

Возвращаемое значение

Объем для открытия короткой позиции.

© 2000-2025, MetaQuotes Ltd.

CheckReverse

Определяет объем для разворота позиции.

```
virtual double  CheckReverse(
   CPositionInfo*  position,     // указатель
   double          sl            // цена Stop Loss
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in] Цена Stop Loss позиции.

Возвращаемое значение

Объем для разворота позиции.

© 2000-2025, MetaQuotes Ltd.

CheckClose

Определяет необходимость закрытия позиции.

```
virtual double  CheckClose()
```

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Модули торговых сигналов

В стандартную поставку клиентского терминала входит набор модулей торговых сигналов для "Мастера MQL5". При создании эксперта в Мастере MQL5, вы можете включить в него любую комбинацию модулей торговых сигналов (до 64). Финальное решение о совершении торговой операции принимается на основе совокупного анализа сигналов всех включенных модулей. Подробное описание механизма принятия решений приведено ниже.

В стандартную поставку входят следующие модули сигналов:

- Сигналы индикатора Accelerator Oscillator
- Сигналы индикатора Adaptive Moving Average
- Сигналы индикатора Awesome Oscillator
- Сигналы осциллятора Bears Power
- Сигналы осциллятора Bulls Power
- Сигналы осциллятора Commodity Channel Index
- Сигналы осциллятора DeMarker
- Сигналы индикатора Double Exponential Moving Average
- Сигналы индикатора Envelopes
- Сигналы индикатора Fractal Adaptive Moving Average
- Сигналы внутридневного временного фильтра
- Сигналы осциллятора MACD
- Сигналы индикатора Moving Average
- Сигналы индикатора Parabolic SAR
- Сигналы осциллятора Relative Strength Index
- Сигналы осциллятора Relative Vigor Index
- Сигналы осциллятора Stochastic
- Сигналы осциллятора Triple Exponential Average
- Сигналы индикатора Triple Exponential Moving Average
- Сигналы осциллятора Williams Percent Range

Механизм принятия торговых решений на основе модулей сигналов

Механизм принятия торговых решений можно представить в виде следующих основных положений:

- Каждый из модулей сигналов обладает своим набором рыночных моделей (определенное сочетание цен и значений индикатора).
- Каждой рыночной модели установлена значимость, измеряемая от 1 до 100. Чем больше значение, тем сильнее модель.
- Каждая из моделей генерирует прогноз движения цены в определенном направлении.
- Прогноз модуля сигналов является результатом поиска заложенных моделей и выдается в виде числа в диапазоне от -100 до +100, где знак определяет направление предполагаемого

© 2000-2025, MetaQuotes Ltd.

движения (отрицательный — цена будет падать, положительный — цена будет расти). Абсолютное значение соответствует силе найденной наилучшей модели.

- Прогноз каждого модуля отправляется на голосование с весовым коэффициентом от 0 до 1.0, указанным в его настройках ("Weight").
- Итогом голосования является число от -100 до +100, где знак определяет направление прогнозируемого движения, а абсолютное значение характеризует силу сигнала. Оно вычисляется как среднеарифметическое взвешенных прогнозов всех модулей сигналов. Данное итоговое значение используется в советнике для принятия торговых решений.

В настройках каждого сгенерированного эксперта присутствуют два параметра — пороговые значения для принятия решения об открытии или закрытии позиции (ThresholdOpen и ThresholdClose), — которые могут иметь значение от 0 до 100. Если сила итогового сигнала (абсолютная величина) преодолевает пороговое значение, принимается решение о совершении торговой операции в направлении, соответствующему знаку прогноза.

Примеры

Пусть существует некий советник с пороговыми значениями ThresholdOpen=20 и ThresholdClose=90. В принятии решений о торговых операциях участвуют модули сигналов на основе MA с весом 0.4 и Stochastic с весом 0.8. Рассмотрим два варианта полученных торговых сигналов:

Вариант 1.

Цена пересекла снизу вверх восходящий индикатор MA. Это соответствует одной из заложенных в модуле MA рыночной модели, предполагающей рост цены. Ее значимость равняется 100. В это же время осциллятор Stochastic развернулся вниз и сформировал дивергенцию с ценой. Это является одной из заложенных в модуле Stochastic моделей, предполагающей падение цены. Значимость этой модели равна 80.

Рассчитаем результат итогового голосования. Взвешенный прогноз, полученный от модуля MA, рассчитывается как 0.4 * 100 = 40. Взвешенный прогноз от модуля Stochastic рассчитывается как 0.8 * (-80) = -64. Итоговый прогноз вычисляется нахождением среднеарифметического этих двух взвешенных прогнозов: (40 - 64)/2 = -12. Это является сигналом на продажу с условной силой 12. Пороговое значение, равное 20, не достигнуто. Соответственно торговая операция не совершается.

Вариант 2.

Цена пересекла сверху вниз восходящий индикатор MA. Это соответствует одной из заложенных в модуле MA рыночной модели, предполагающей рост цены. Ее значимость равняется 10. В это же время осциллятор Stochastic развернулся вниз и сформировал дивергенцию с ценой. Это является одной из заложенных в модуле Stochastic моделей, предполагающей падение цены. Значимость этой модели равна 80.

Рассчитаем результат итогового голосования. Взвешенный прогноз, полученный от модуля MA, рассчитывается как 0.4 * 10 = 4. Взвешенный прогноз от модуля Stochastic рассчитывается как 0.8 * (-80) = -64. Итоговый прогноз вычисляется нахождением среднеарифметического этих двух взвешенных прогнозов: (4 - 64)/2 = -30. Это является сигналом на продажу с условной силой 30. Пороговое значение, равное 20, достигнуто. Соответственно, результатом является сигнал на открытие короткой позиции.

© 2000-2025, MetaQuotes Ltd.



a) Дивергенция цены и осциллятора Stochastic (вариант 1 и 2).

b) Цена пересекла индикатор MA снизу вверх (вариант 1).

c) Цена пересекла индикатор MA сверху вниз (вариант 2).

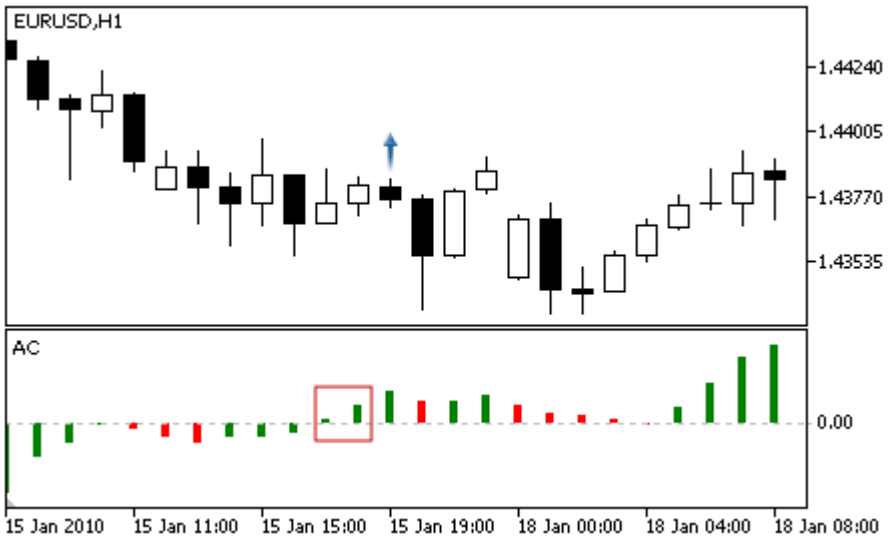
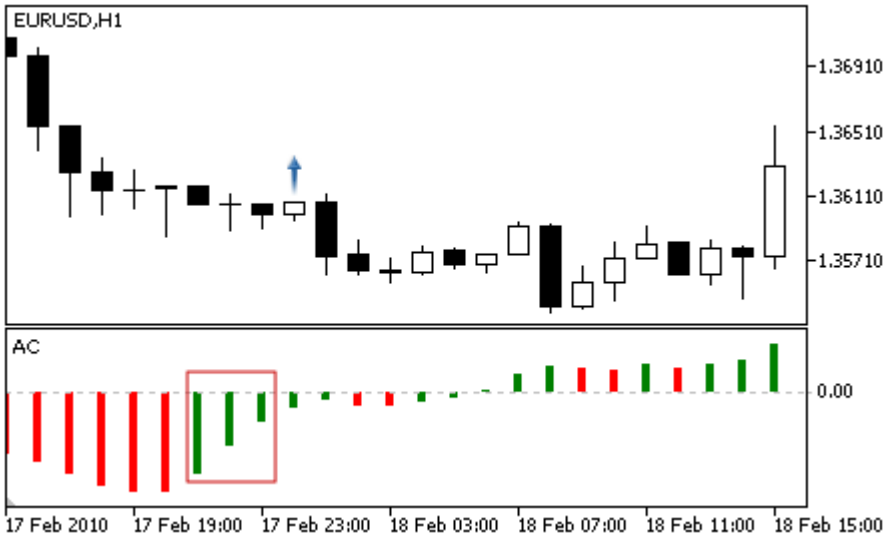
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Accelerator Oscillator

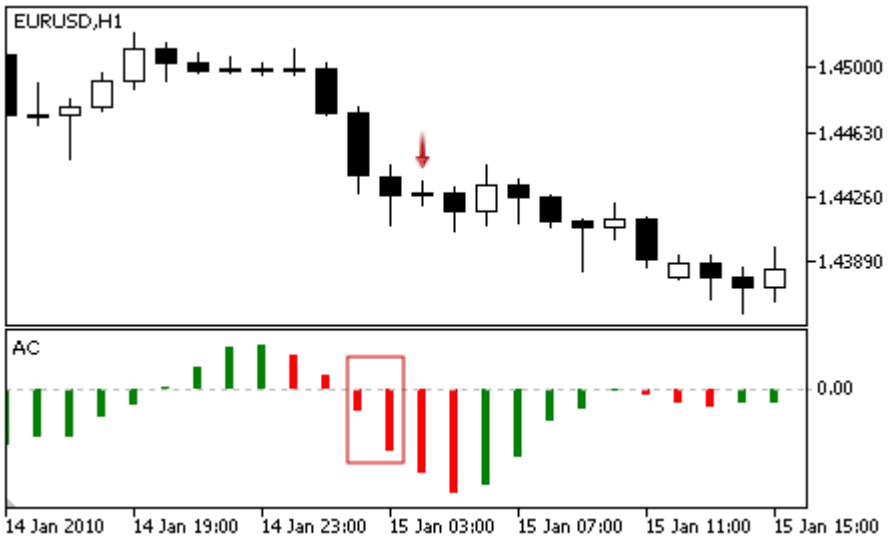
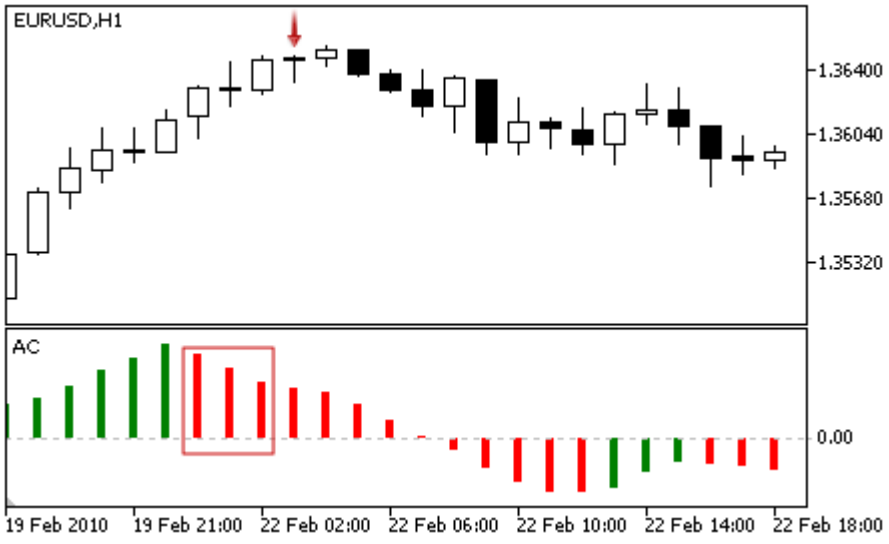
Данный модуль сигналов основан на рыночных моделях индикатора Accelerator Oscillator. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• Значение индикатора выше 0 и оно растет на анализируемом баре и предыдущем.  • Значение индикатора ниже 0 и оно растет на анализируемом баре и двух предыдущих. 
За продажу	• Значение индикатора ниже 0 и оно падает на анализируемом баре и предыдущем.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • Значение индикатора выше 0 и оно падает на анализируемом баре и двух предыдущих. 
Не против покупки	Значение индикатора на анализируемом баре растет.
Не против продажи	Значение индикатора на анализируемом баре падает.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.

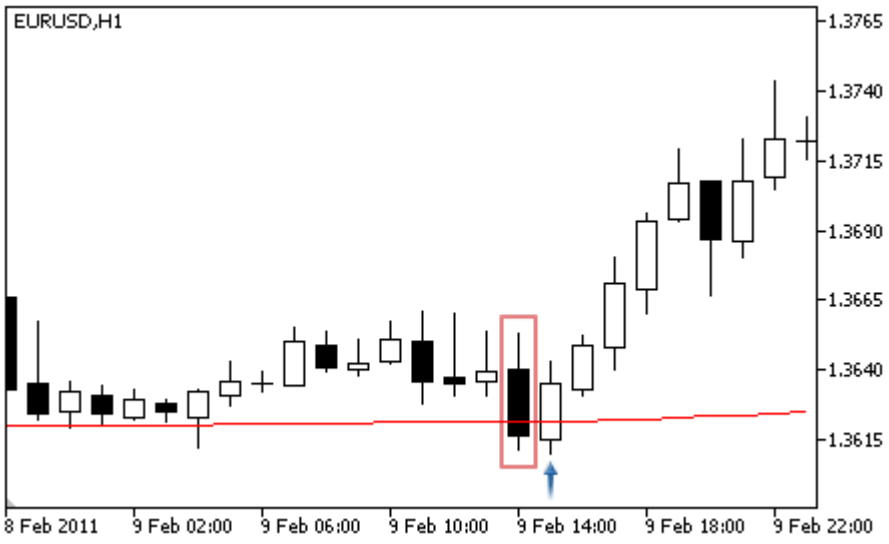
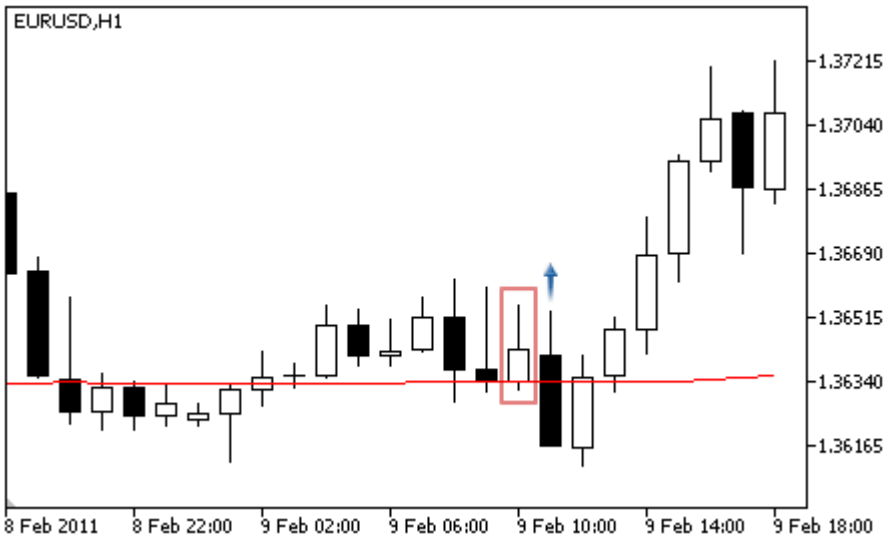
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Adaptive Moving Average

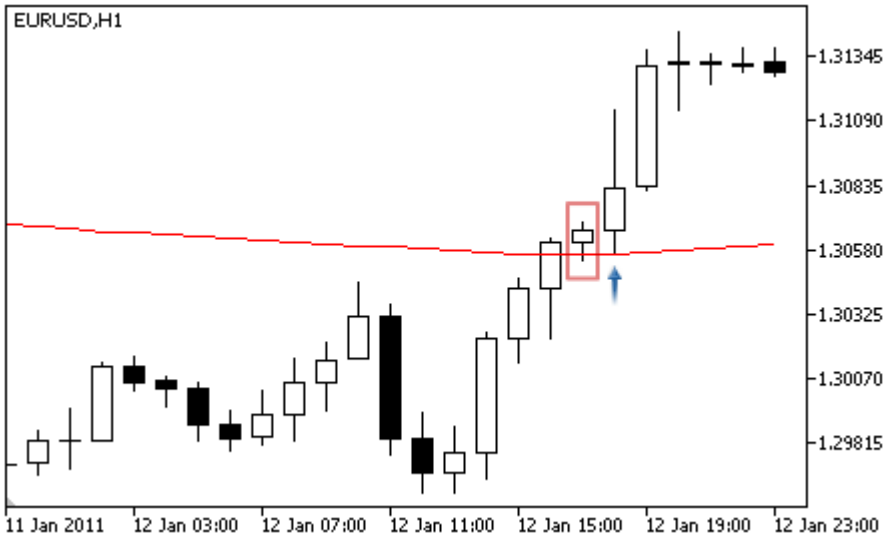
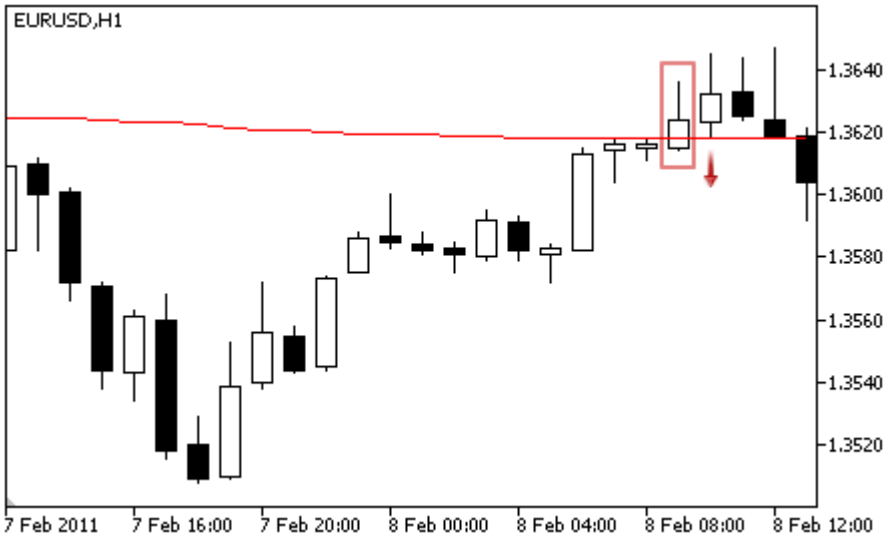
Данный модуль сигналов основан на рыночных моделях индикатора Adaptive Moving Average. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

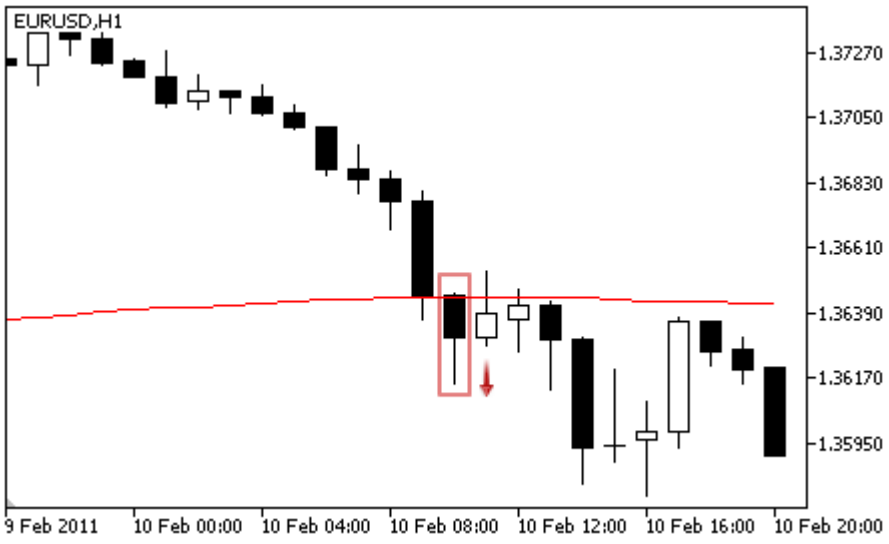
Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

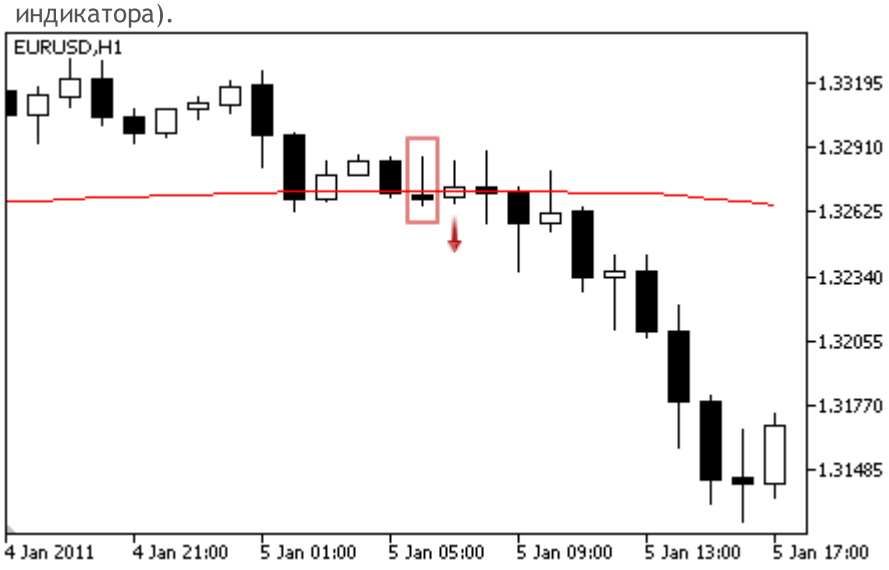
Тип сигнала	Описание условий
За покупку	• **Несформировавшийся прокол.** Цена пересекла индикатор сверху вниз(цена Open анализируемого бара выше линии индикатора, а цена Close - ниже), но индикатор растет (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше) и индикатор растет (сильный сигнал). 

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	• **Сформировашийся прокол.** Цена пересекла индикатор нижней тенью (цены Open и Close анализируемого бара выше линии индикатора, а цена Low ниже) и индикатор растет (сигнал на отбой от линии индикатора). 
За продажу	• **Несформировавшийся прокол.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше), но индикатор падает (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор сверху вниз (цена Open анализируемого бара выше линии индикатора, а цена Close - ниже) и индикатор падает (сильный сигнал).

© 2000-2025, MetaQuotes Ltd.

<table>
<tr><th>Тип сигнала</th><th>Описание условий</th></tr>
<tr><td></td><td> 

Сформировашийся прокол. Цена пересекла индикатор верхней тенью (цены Open и Close анализируемого бара ниже линии индикатора, а цена High выше) и индикатор падает (сигнал на отбой от линии индикатора).

</td></tr>
<tr><td>Не против покупки</td><td>Цена находится выше индикатора.</td></tr>
<tr><td>Не против продажи</td><td>Цена находится ниже индикатора.</td></tr>
</table>

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodMA	Период усреднения индикатора.
Shift	Смещение индикатора по оси времени (в барах).
Method	Метод усреднения.
Applied	Ценовой ряд, на значениях которого рассчитывается индикатор.

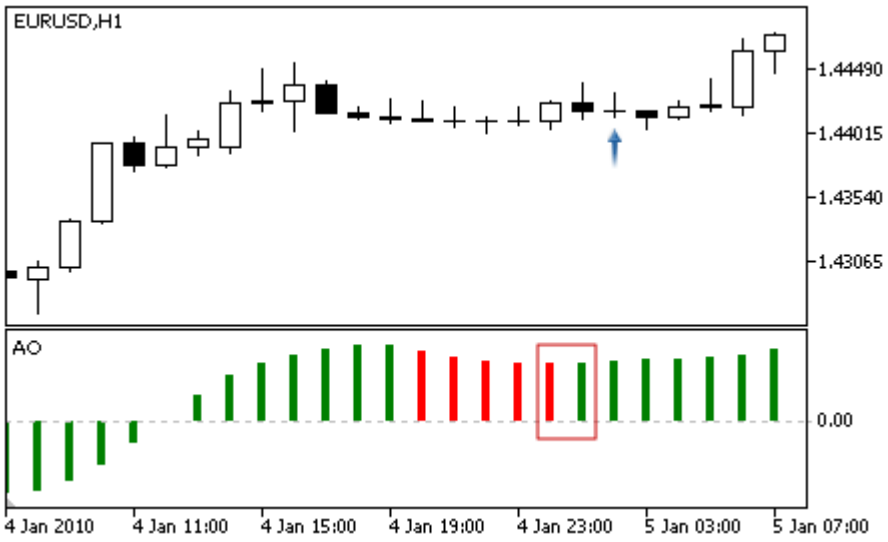
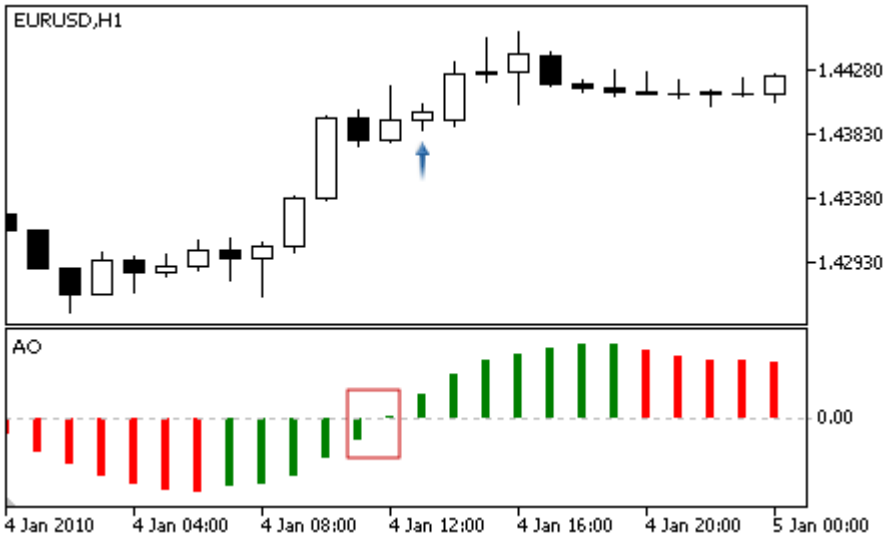
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Awesome Oscillator

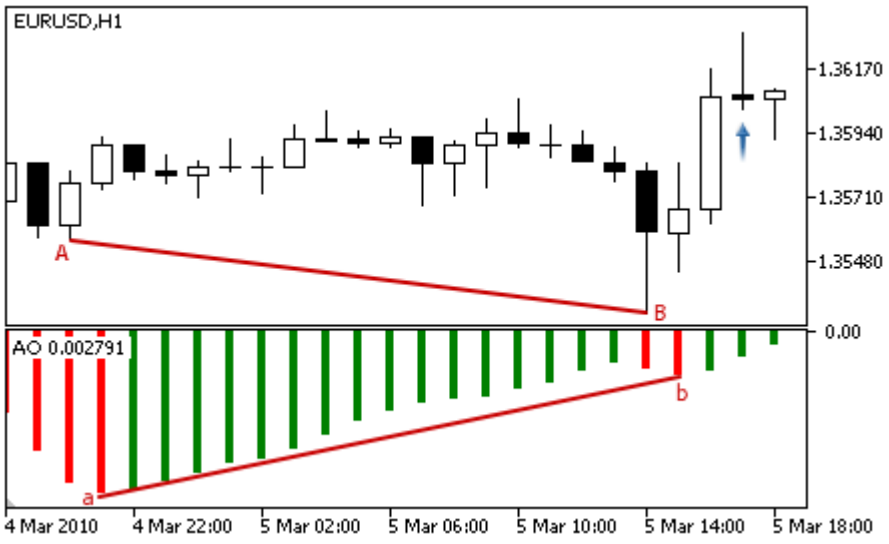
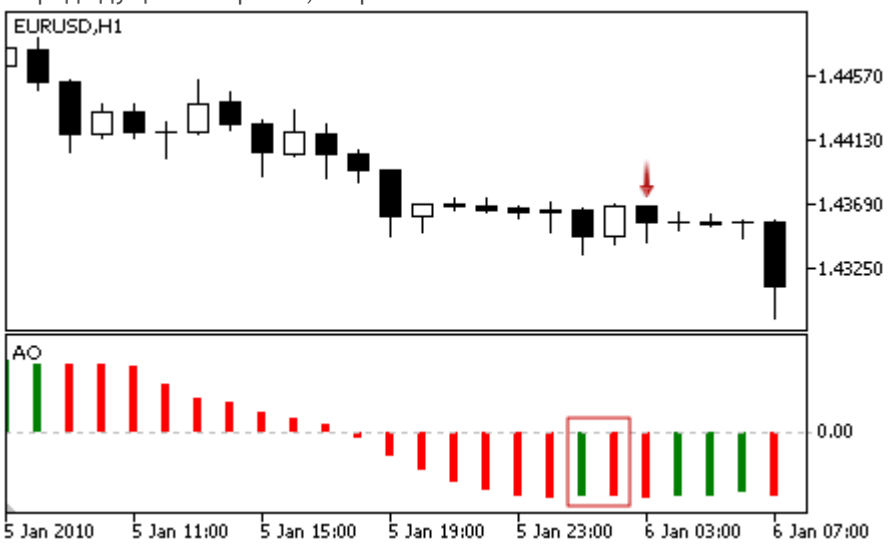
Данный модуль сигналов основан на рыночных моделях индикатора Awesome Oscillator. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

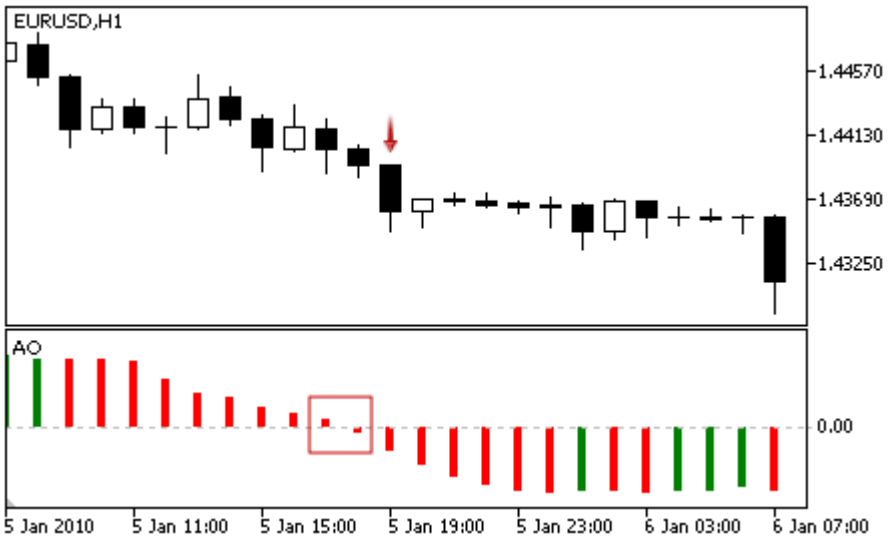
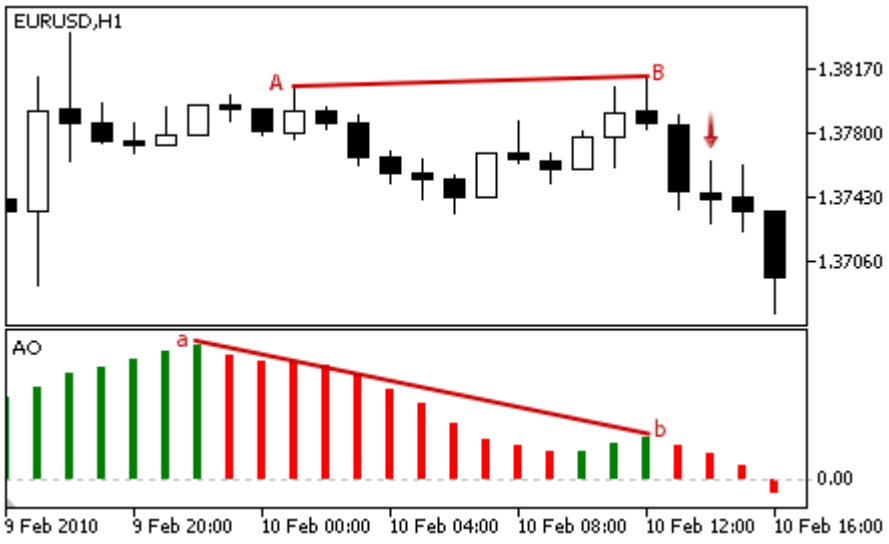
Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Блюдце** — значение индикатора на анализируемом баре растет, а на предыдущем оно падало, и при этом оба значения выше 0.  • **Пересечение нулевой линии** — значение индикатора на анализируемом баре больше 0, а на предыдущем - меньше 0.  • **Дивергенция** — первая анализируемая впадина индикатора мельче предыдущей, а соответствующая ей впадина цены глубже предыдущей. При этом индикатор не должен подниматься выше нулевого уровня.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	
За продажу	• **Блюдце** — значение индикатора на анализируемом баре падает, а на предыдущем оно росло, и при этом оба значения ниже 0.  • **Пересечение нулевой линии** — значение индикатора на анализируемом баре меньше 0, а на предыдущем - больше 0.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Дивергенция** — первый анализируемый пик индикатора ниже предыдущего, а соответствующий ему пик цены выше предыдущего. При этом индикатор не должен опускаться ниже нулевого уровня. 
Не против покупки	Значение индикатора на анализируемом баре растет.
Не против продажи	Значение индикатора на анализируемом баре падает.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.

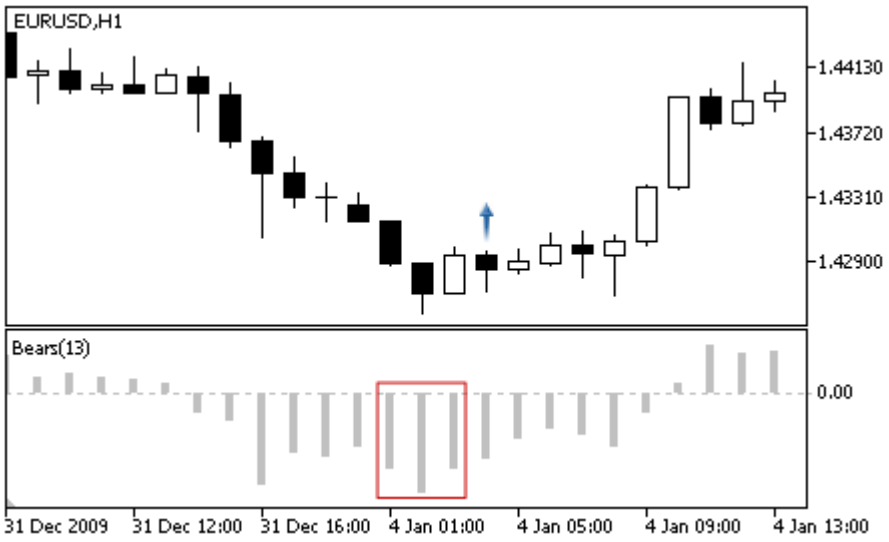
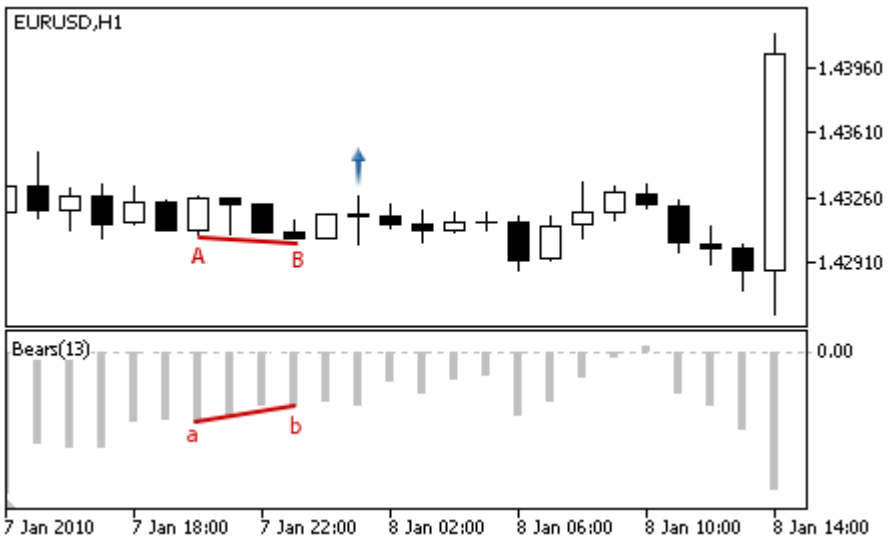
© 2000-2025, MetaQuotes Ltd.

Сигналы осциллятора Bears Power

Данный модуль сигналов основан на рыночных моделях осциллятора Bears Power. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Разворот** — осциллятор развернулся вверх и при этом его значение на анализируемом баре ниже 0.  • **Дивергенция** — первая анализируемая впадина осциллятора мельче предыдущей, а соответствующая ей впадина цены глубже предыдущей. При этом осциллятор не должен подниматься выше нулевого уровня. 
За продажу	Сигналы для продажи отсутствуют.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
Не против покупки	Значение осциллятора ниже 0.
Не против продажи	Сигналы отсутствуют.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodBears	Период расчета осциллятора.

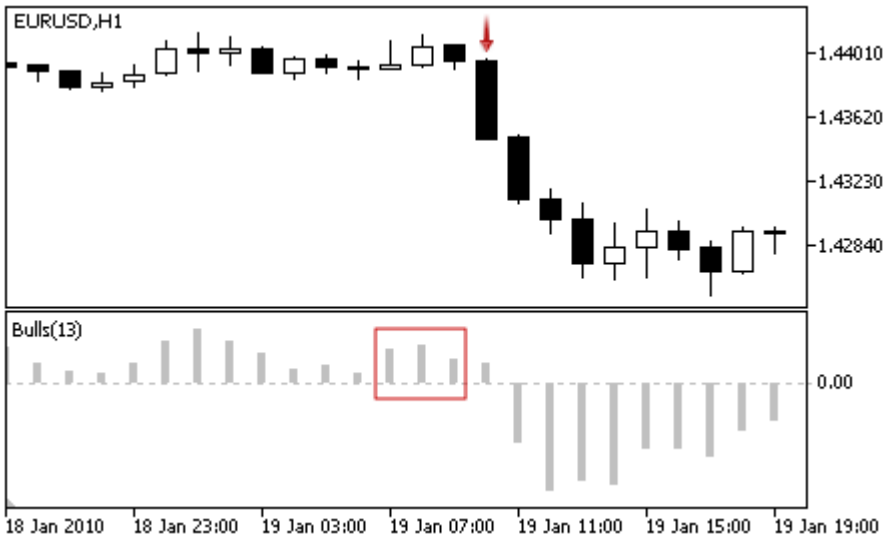
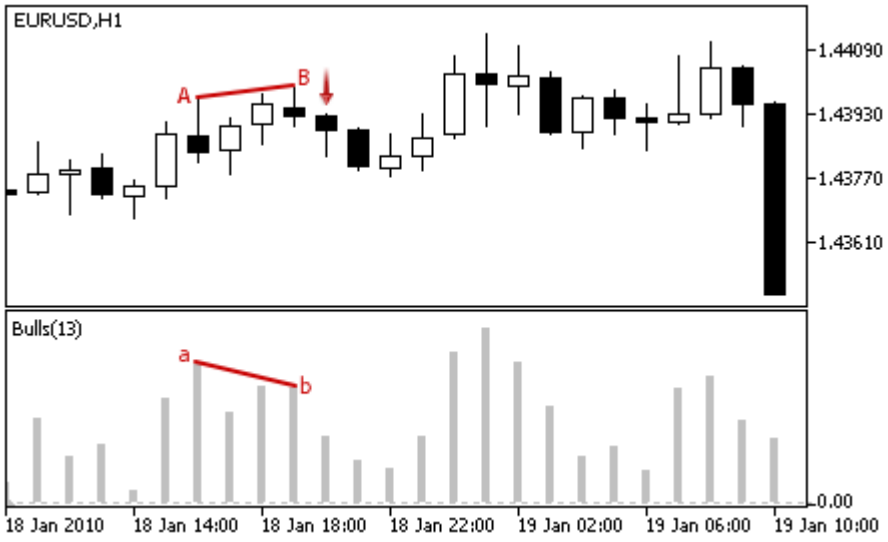
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Bulls Power

Данный модуль сигналов основан на рыночных моделях осциллятора Bulls Power. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	Сигналы для покупки отсутствуют.
За продажу	• **Разворот** — осциллятор развернулся вниз и при этом его значение на анализируемом баре выше 0.  • **Дивергенция** — первый анализируемый пик осциллятора ниже предыдущего, а соответствующий ему пик цены выше предыдущего. При этом осциллятор не должен опускаться ниже нулевого уровня. 
Не против покупки	Сигналы отсутствуют.
Не против продажи	Значение осциллятора выше 0.

© 2000-2025, MetaQuotes Ltd.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodBulls	Период расчета осциллятора.

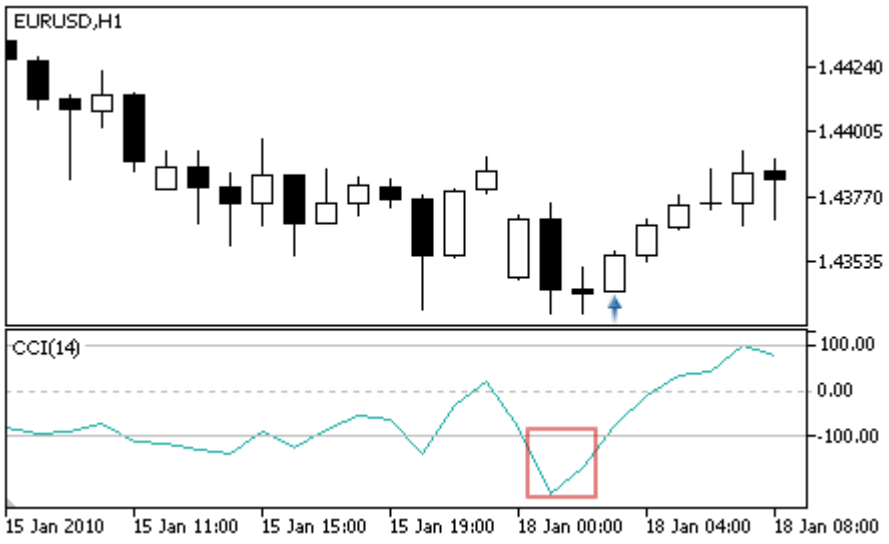
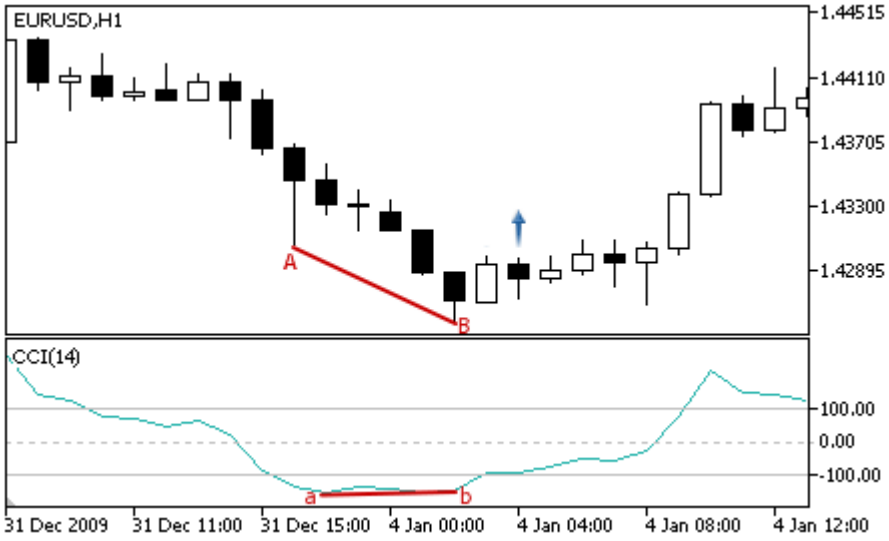
© 2000-2025, MetaQuotes Ltd.

Сигналы осциллятора Commodity Channel Index

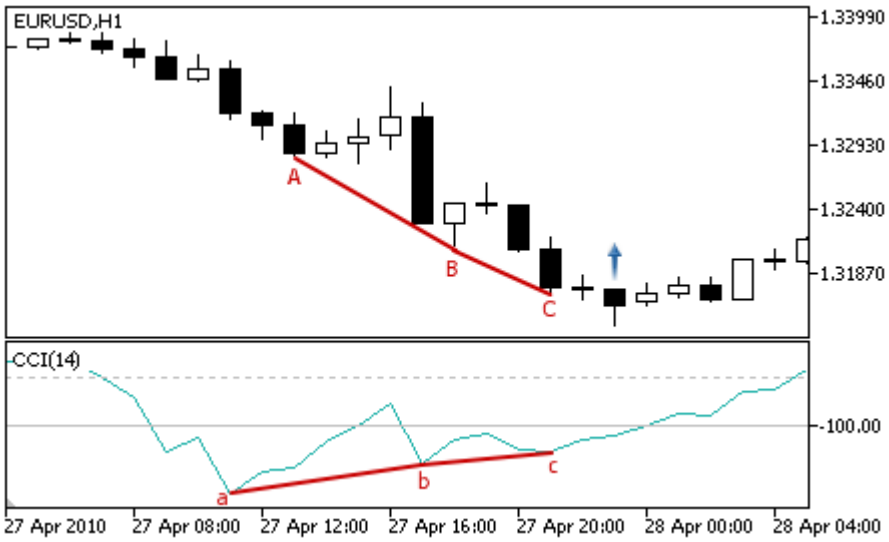
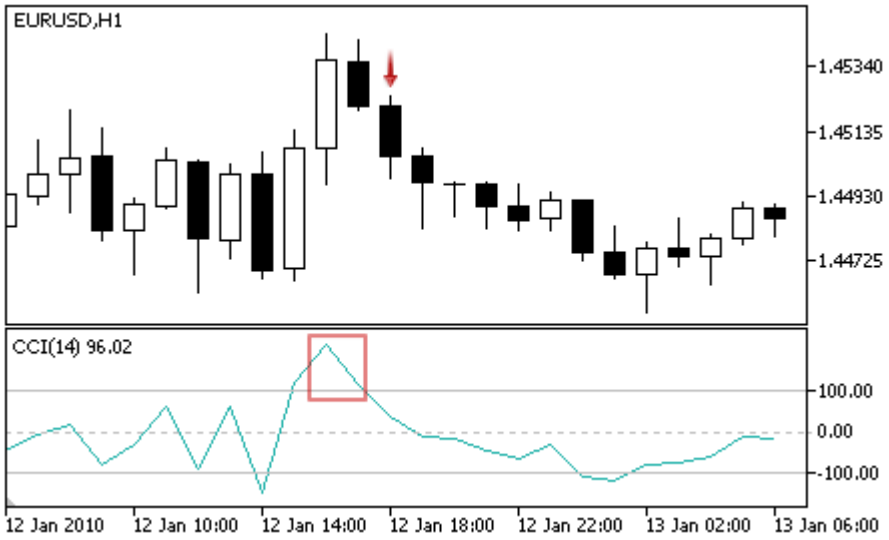
Данный модуль сигналов основан на рыночных моделях осциллятора Commodity Channel Index. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

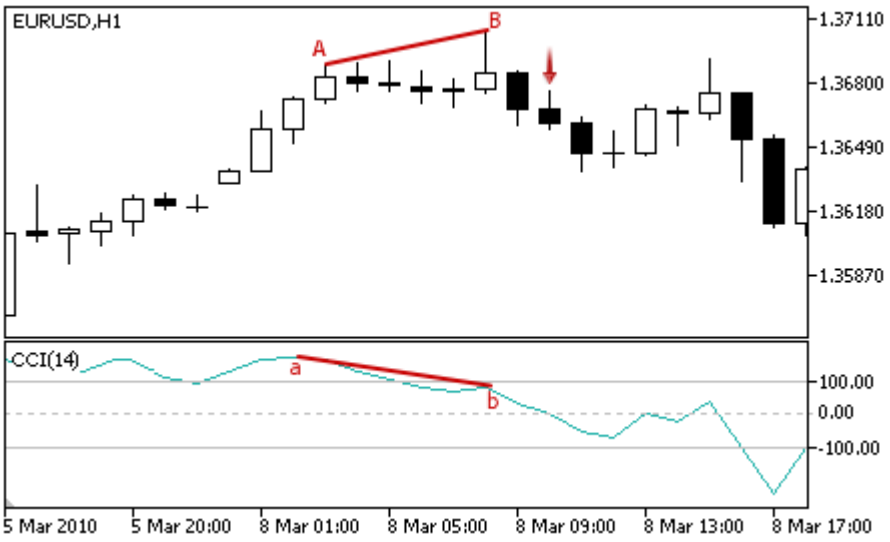
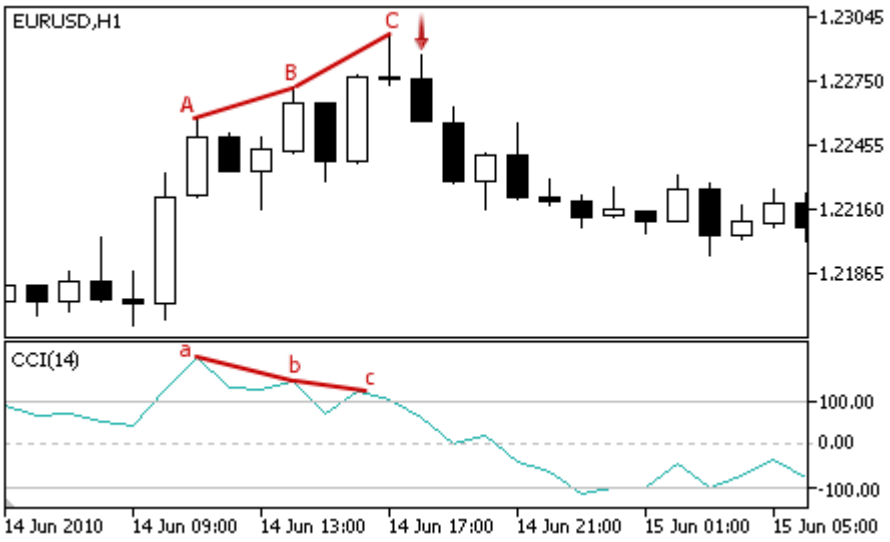
Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Разворот за уровнем перепроданности** — осциллятор развернулся вверх и его значение на анализируемом баре находится за уровнем перепроданности (по умолчанию -100).  • **Дивергенция** — первая анализируемая впадина осциллятора мельче предыдущей, а соответствующая ей впадина цены глубже предыдущей.  • **Двойная дивергенция** — осциллятор сформировал три последовательных впадины, каждая из которых мельче предыдущей,

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	а цена сформировала три соответствующие им впадины, каждая из которых глубже предыдущей. 
За продажу	• **Разворот за уровнем перекупленности** — осциллятор развернулся вниз и его значение на анализируемом баре находится за уровнем перекупленности (по умолчанию 100).  • **Дивергенция** — первый анализируемый пик осциллятора ниже предыдущего, а соответствующий ему пик цены выше предыдущего.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Двойная дивергенция** — осциллятор сформировал три последовательных пика, каждый из которых ниже предыдущего, а цена сформировала три соответствующих им пика, каждый из которых выше предыдущего. 
Не против покупки	Значение осциллятора на анализируемом баре растет.
Не против продажи	Значение осциллятора на анализируемом баре падает.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodCCI	Период расчета осциллятора.
Applied	Ценовой ряд, на значениях которого рассчитывается осциллятор.

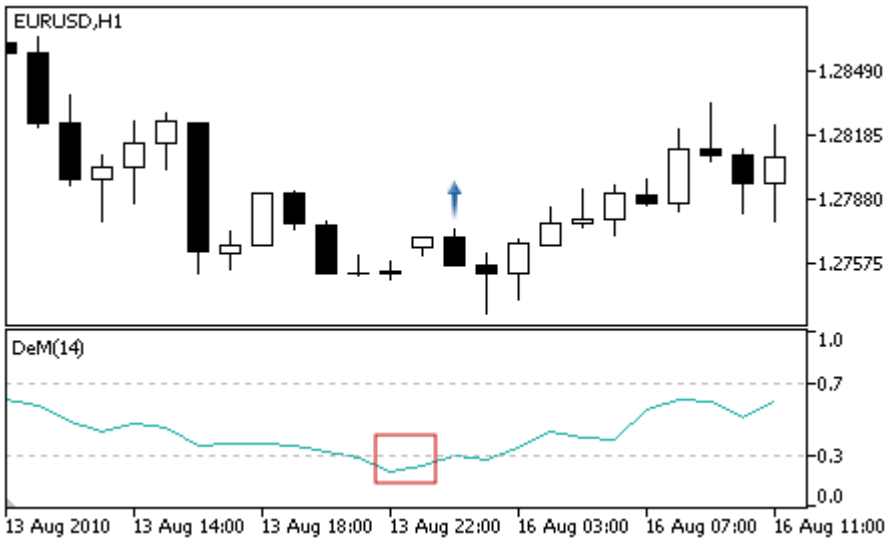
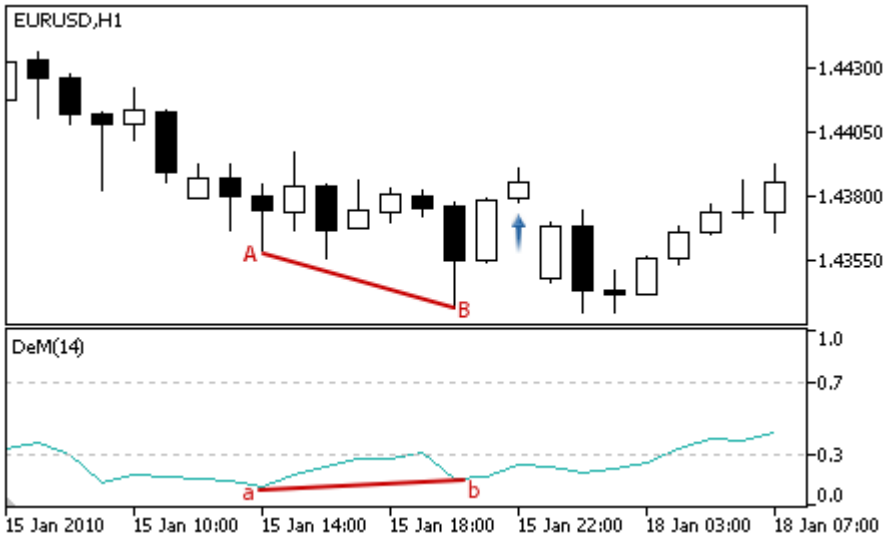
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора DeMarker

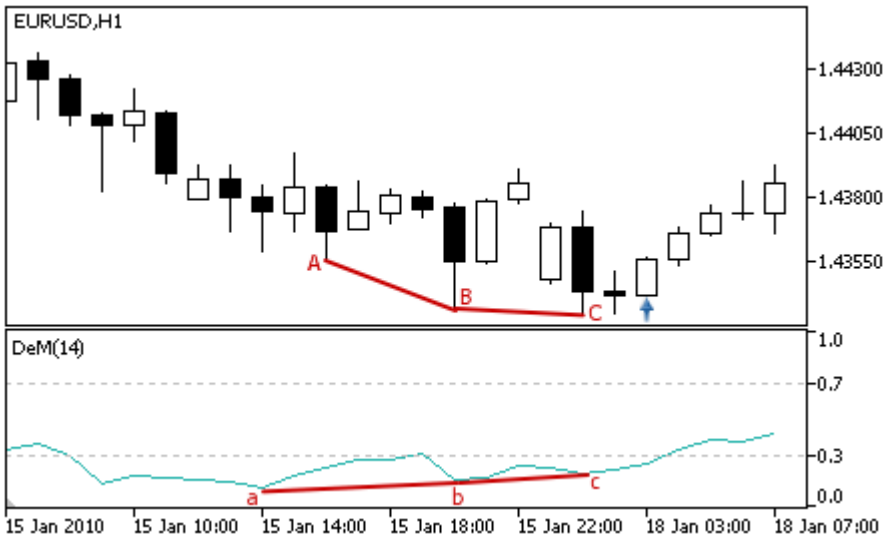
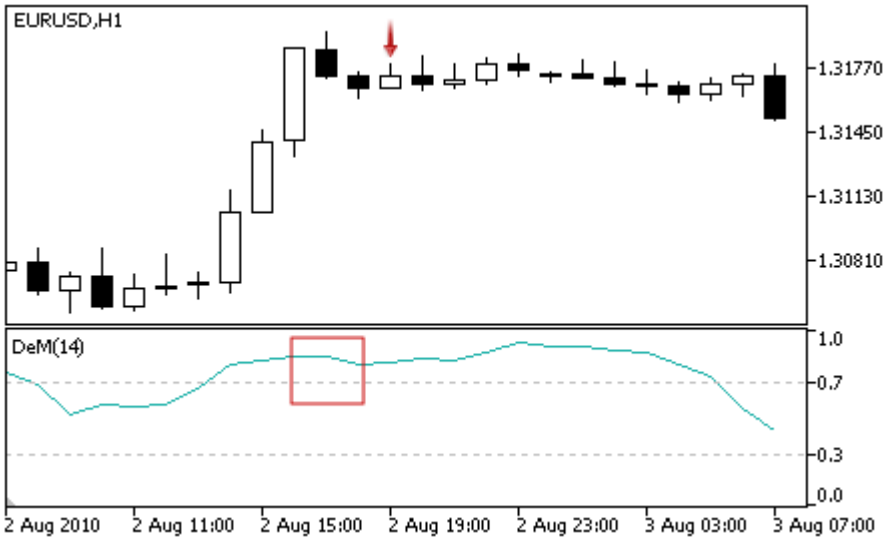
Данный модуль сигналов основан на рыночных моделях осциллятора DeMarker. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

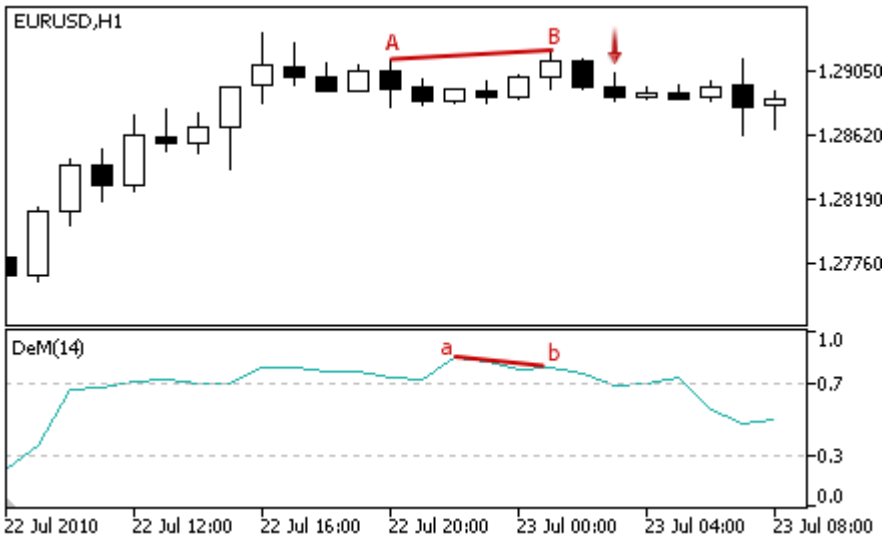
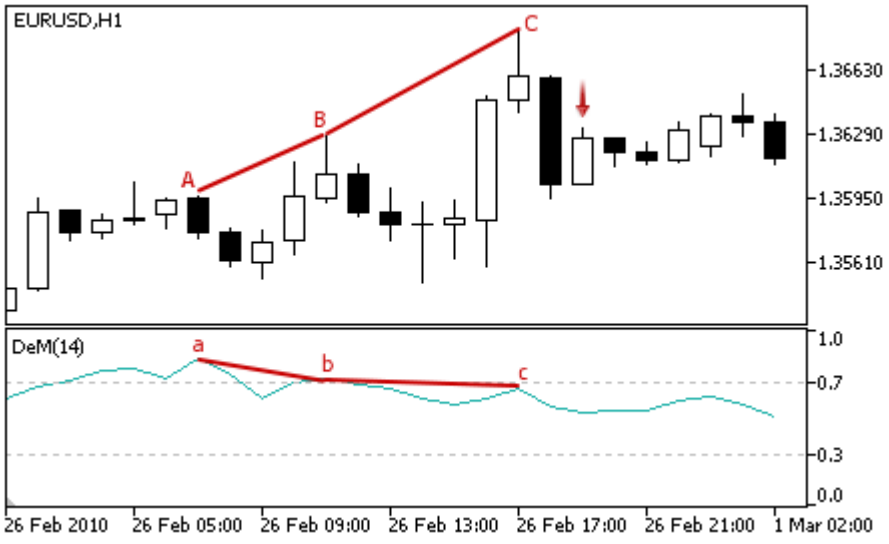
Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Разворот за уровнем перепроданности** — осциллятор развернулся вверх и его значение на анализируемом баре находится за уровнем перепроданности (по умолчанию 0.3).  • **Дивергенция** — первая анализируемая впадина осциллятора мельче предыдущей, а соответствующая ей впадина цены глубже предыдущей.  • **Двойная дивергенция** — осциллятор сформировал три последовательных впадины, каждая из которых мельче предыдущей,

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	а цена сформировала три соответствующие им впадины, каждая из которых глубже предыдущей. 
За продажу	• **Разворот за уровнем перекупленности** — осциллятор развернулся вниз и его значение на анализируемом баре находится за уровнем перекупленности (по умолчанию 0.7).  • **Дивергенция** — первый анализируемый пик осциллятора ниже предыдущего, а соответствующий ему пик цены выше предыдущего.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Двойная дивергенция** — осциллятор сформировал три последовательных пика, каждый из которых ниже предыдущего, а цена сформировала три соответствующих им пика, каждый из которых выше предыдущего. 
Не против покупки	Значение осциллятора на анализируемом баре растет.
Не против продажи	Значение осциллятора на анализируемом баре падает.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodDeM	Период расчета осциллятора.

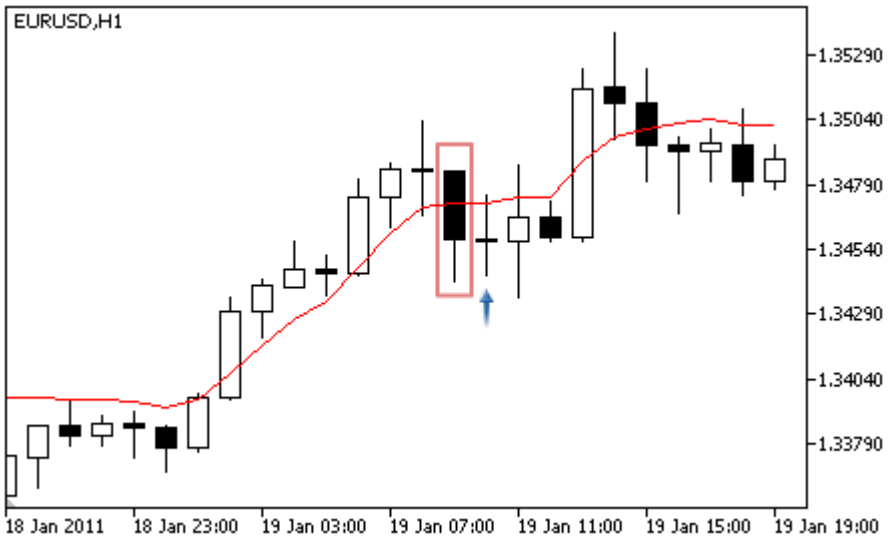
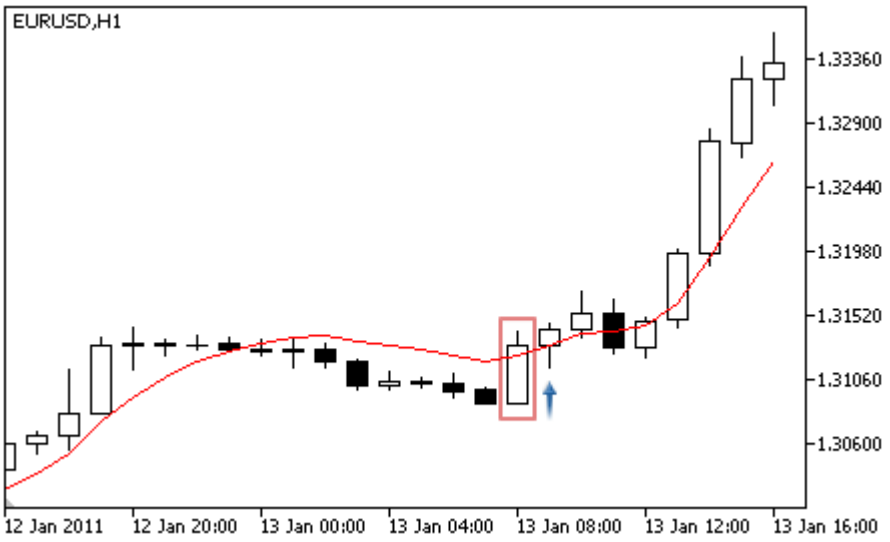
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Double Exponential Moving Average

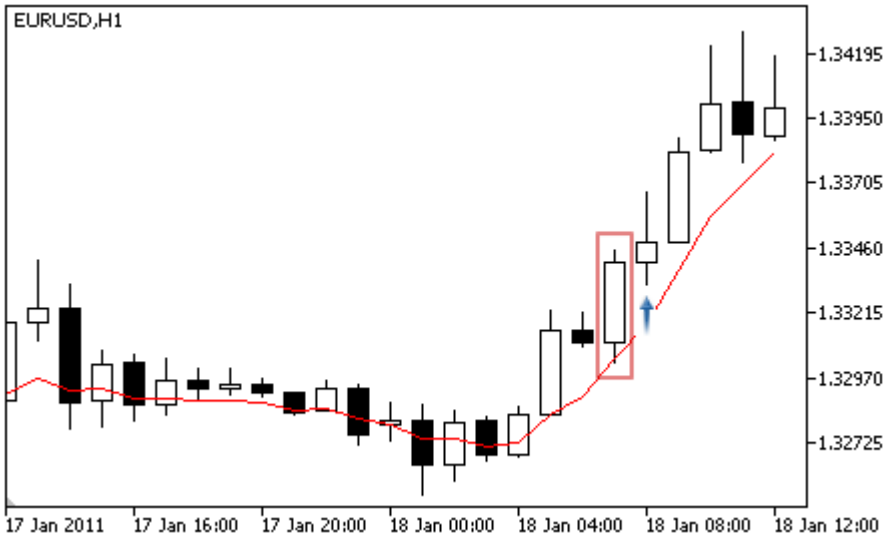
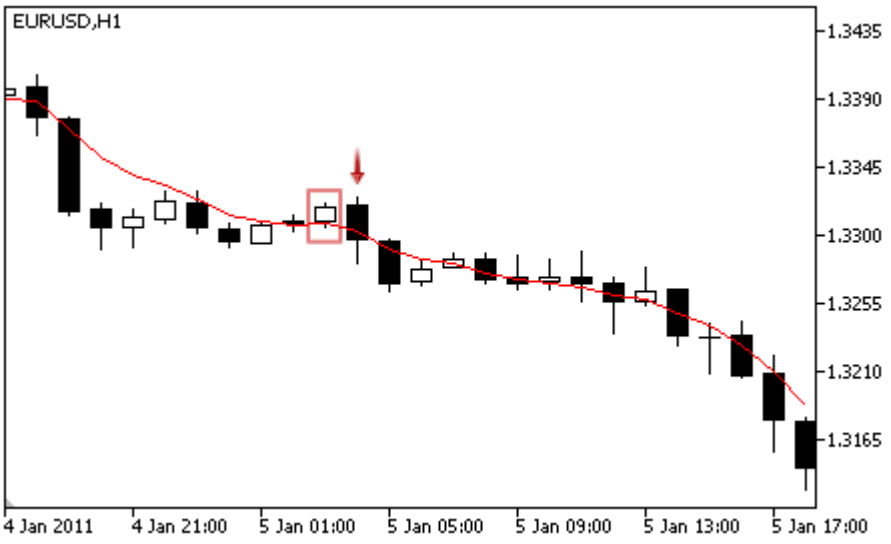
Данный модуль сигналов основан на рыночных моделях индикатора Double Exponential Moving Average. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

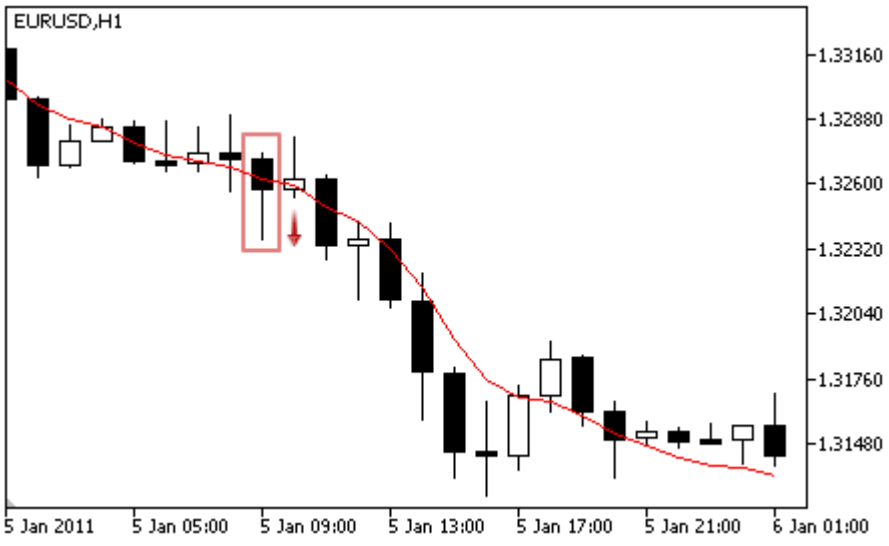
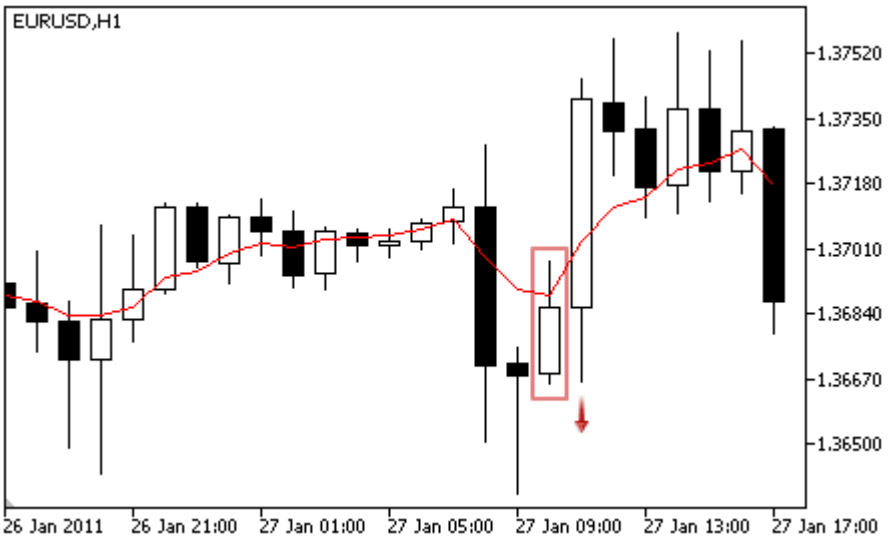
Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Несформировавшийся прокол.** Цена пересекла индикатор сверху вниз(цена Open анализируемого бара выше линии индикатора, а цена Close - ниже), но индикатор растет (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше) и индикатор растет (сильный сигнал). 

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	• **Сформировашийся прокол.** Цена пересекла индикатор нижней тенью (цены Open и Close анализируемого бара выше линии индикатора, а цена Low ниже) и индикатор растет (сигнал на отбой от линии индикатора). 
За продажу	• **Несформировавшийся прокол.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше), но индикатор падает (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор сверху вниз (цена Open анализируемого бара выше линии индикатора, а цена Close - ниже) и индикатор падает (сильный сигнал).

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Сформировашийся прокол.** Цена пересекла индикатор верхней тенью (цены Open и Close анализируемого бара ниже линии индикатора, а цена High выше) и индикатор падает (сигнал на отбой от линии индикатора). 
Не против покупки	Цена находится выше индикатора.
Не против продажи	Цена находится ниже индикатора.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodMA	Период усреднения индикатора.
Shift	Смещение индикатора по оси времени (в барах).
Method	Метод усреднения.
Applied	Ценовой ряд, на значениях которого рассчитывается индикатор.

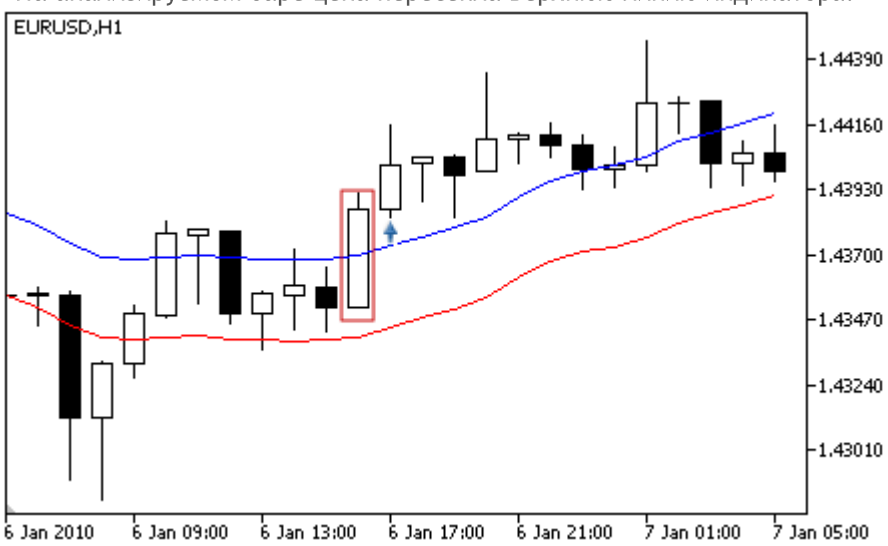
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Envelopes

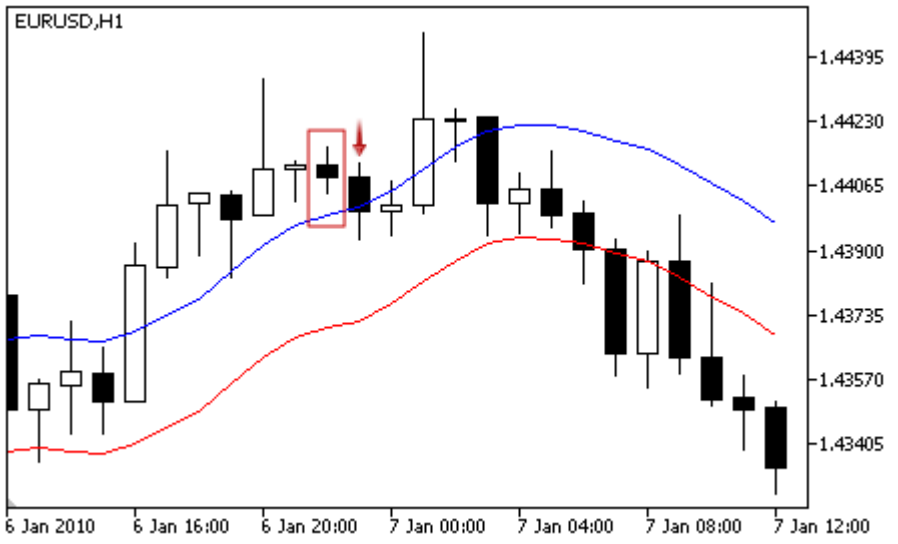
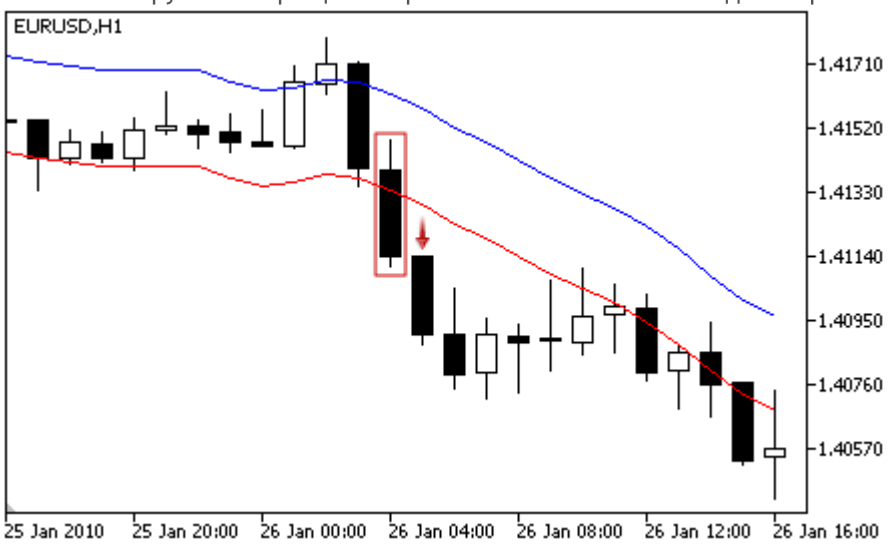
Данный модуль сигналов основан на рыночных моделях индикатора Envelopes. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• На анализируемом баре цена находится вблизи нижней линии индикатора.  • На анализируемом баре цена пересекла верхнюю линию индикатора. 
За продажу	• На анализируемом баре цена находится вблизи верхней линии индикатора.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • На анализируемом баре цена пересекла нижнюю линию индикатора. 
Не против покупки	Сигналы отсутствуют.
Не против продажи	Сигналы отсутствуют.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodMA	Период расчет индикатора.
Shift	Смещение индикатора по оси времени (в барах).
Method	Метод усреднения.
Applied	Ценовой ряд, на значениях которого рассчитывается индикатор.
Deviation	Отклонение границ конверта от центральной линии (MA) в процентах.

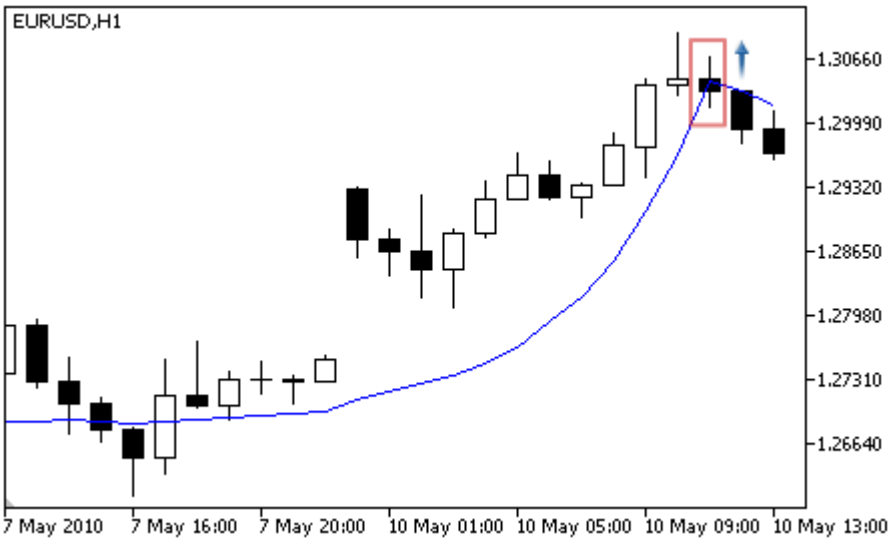
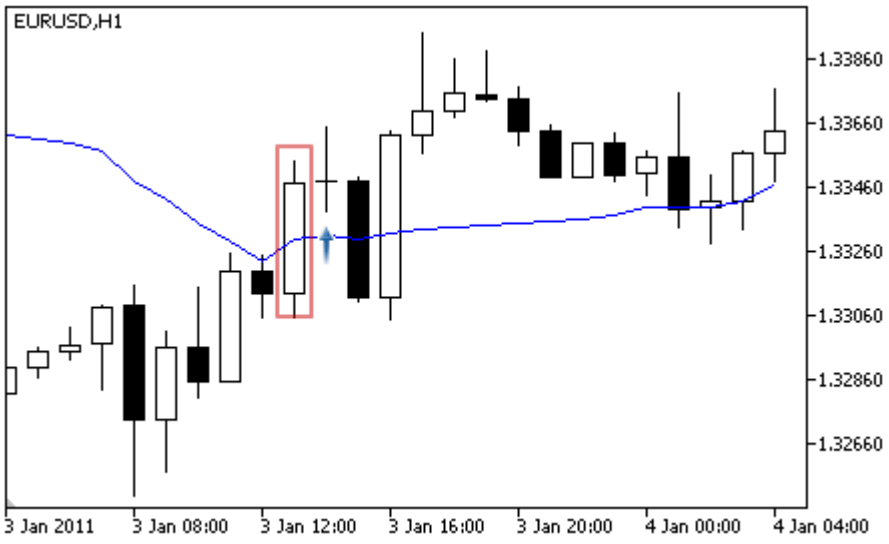
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Fractal Adaptive Moving Average

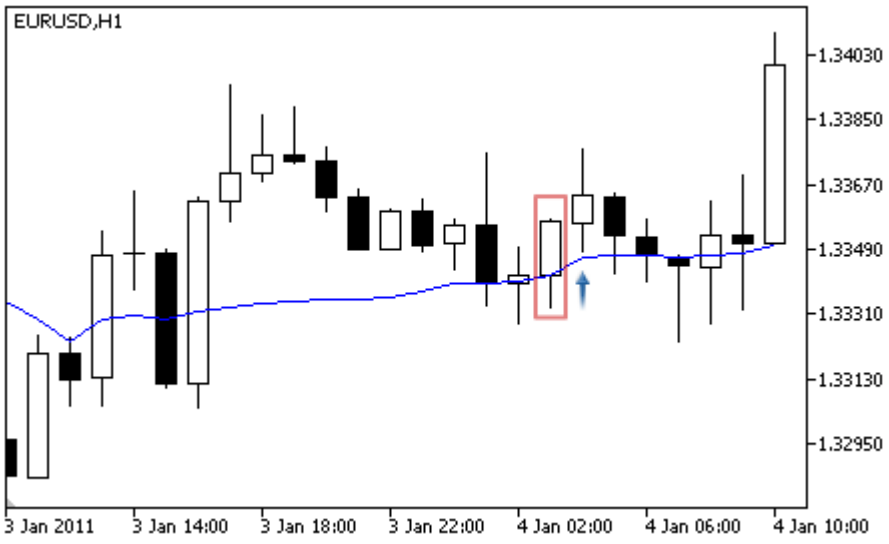
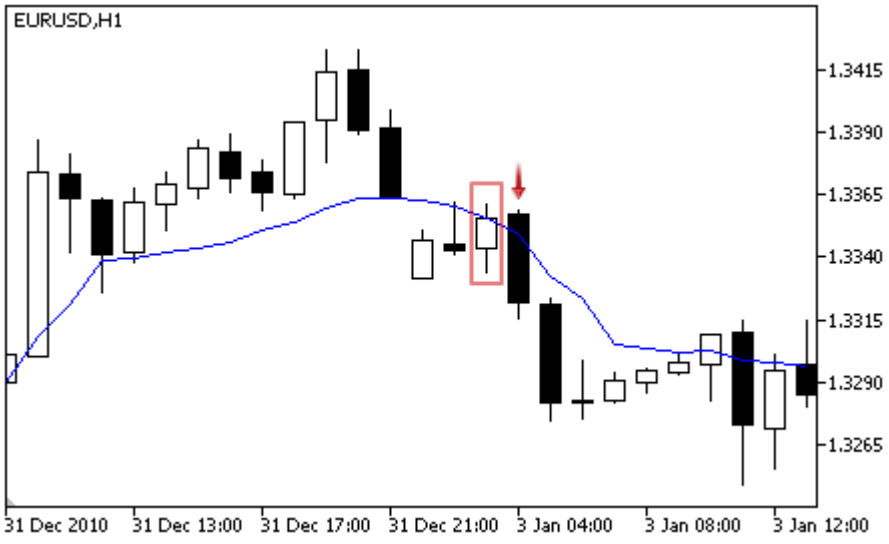
Данный модуль сигналов основан на рыночных моделях индикатора Fractal Adaptive Moving Average. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

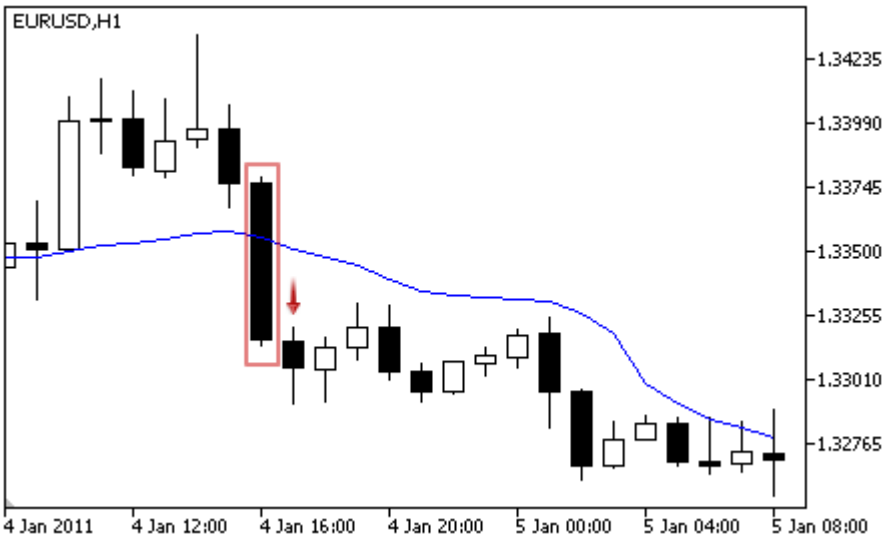
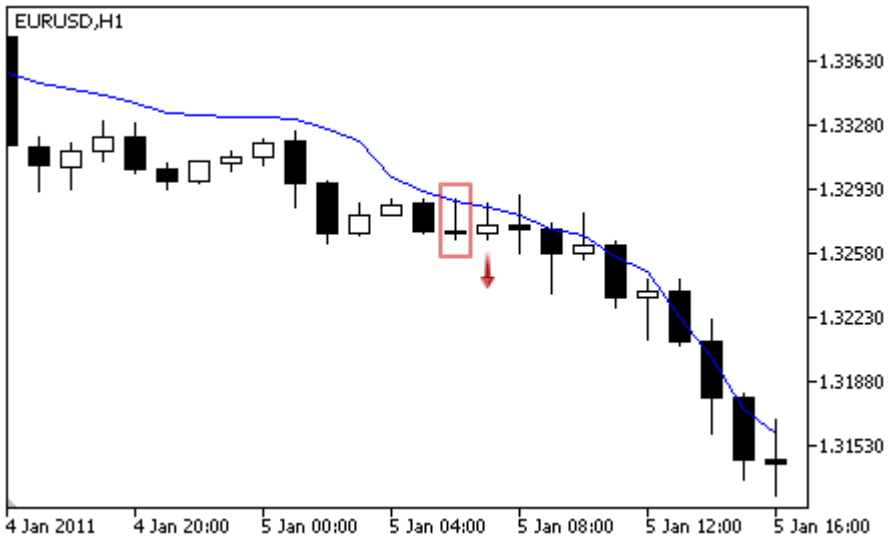
Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Несформировавшийся прокол.** Цена пересекла индикатор сверху вниз(цена Open анализируемого бара выше линии индикатора, а цена Close - ниже), но индикатор растет (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше) и индикатор растет (сильный сигнал). 

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	• **Сформировашийся прокол.** Цена пересекла индикатор нижней тенью (цены Open и Close анализируемого бара выше линии индикатора, а цена Low ниже) и индикатор растет (сигнал на отбой от линии индикатора). 
За продажу	• **Несформировавшийся прокол.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше), но индикатор падает (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор сверху вниз (цена Open анализируемого бара выше линии индикатора, а цена Close - ниже) и индикатор падает (сильный сигнал).

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Сформировашийся прокол.** Цена пересекла индикатор верхней тенью (цены Open и Close анализируемого бара ниже линии индикатора, а цена High выше) и индикатор падает (сигнал на отбой от линии индикатора). 
Не против покупки	Цена находится выше индикатора.
Не против продажи	Цена находится ниже индикатора.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodMA	Период усреднения индикатора.
Shift	Смещение индикатора по оси времени (в барах).
Method	Метод усреднения.
Applied	Ценовой ряд, на значениях которого рассчитывается индикатор.

© 2000-2025, MetaQuotes Ltd.

Сигналы внутридневного временного фильтра

Данный модуль сигналов основан на том, что эффективность рыночных моделей меняется во времени. Используя данный модуль, можно отфильтровывать сигналы, получаемые от других модулей, по часам и дням недели. Это позволяет улучшить качество генерируемых сигналов за счет отсечения неблагоприятных участков времени. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	Сигналы отсутствуют.
За продажу	Сигналы отсутствуют.
Не против покупки	Текущее время и день недели удовлетворяют заданным параметрам.
Не против продажи	Текущее время и день недели удовлетворяют заданным параметрам.

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
GoodHourOfDay	Номер единственного часа в сутках (от 0 до 23), в который будут разрешены торговые сигналы. При значении -1 сигналы разрешены круглосуточно.
BadHoursOfDay	Битовое поле, каждый бит которого соответствует часу в сутках (0 бит - 0 час, ..., 23 бит - 23 час). Если значение бита равно 0, торговые сигналы в соответствующем часе разрешены. Если значение бита равно 1, торговые сигналы в соответствующем часе запрещены. Указанное число представляется в виде двоичного и используется в виде битовой маски. Запрещенные часы имеют приоритет над разрешенными.
GoodDayOfWeek	Номер единственного дня недели (от 0 до 6, где 0 - воскресенье), в который будут разрешены торговые сигналы. При значении -1 сигналы разрешены в любой день недели.
BadDaysOfWeek	Битовое поле, каждый бит которого соответствует дню недели (0 бит - воскресенье, ..., 6 бит - суббота). Если значение бита равно 0, торговые сигналы в соответствующий день разрешены. Если значение бита равно 1, торговые сигналы в соответствующем день запрещены. Указанное число представляется в виде двоичного и используется в виде битовой маски. Запрещенные дни имеют приоритет над разрешенными.

© 2000-2025, MetaQuotes Ltd.

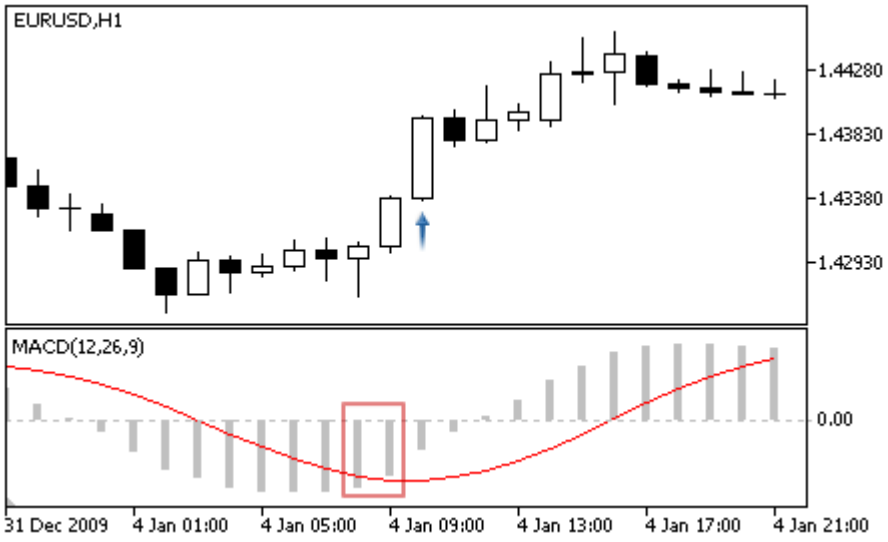
© 2000-2025, MetaQuotes Ltd.

Сигналы осциллятора MACD

Данный модуль сигналов основан на рыночных моделях осциллятора MACD. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

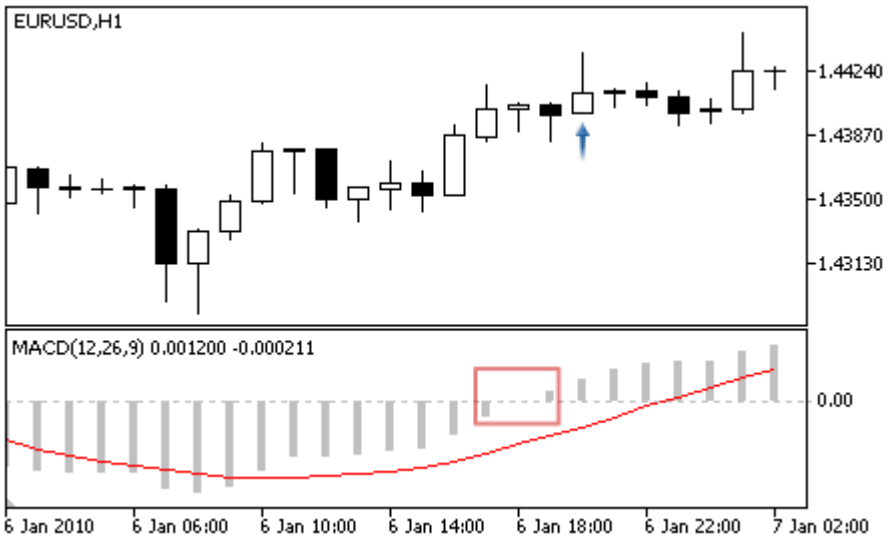
Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

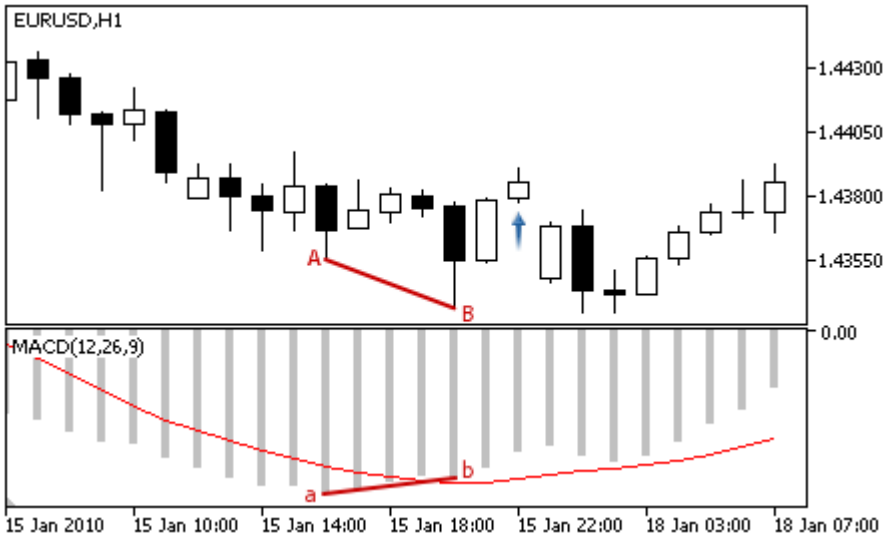
Тип сигнала	Описание условий
За покупку	• **Разворот** — осциллятор развернулся вверх (осциллятор растет на анализируемом баре, а на предыдущем он падал).  • **Пересечение основной и сигнальной линии** — на анализируемом баре основная линия выше сигнальной, а на предыдущем — ниже.  • **Пересечении нулевого уровня** — на анализируемом баре основная линия выше нулевого уровня, а на предыдущем — ниже.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий

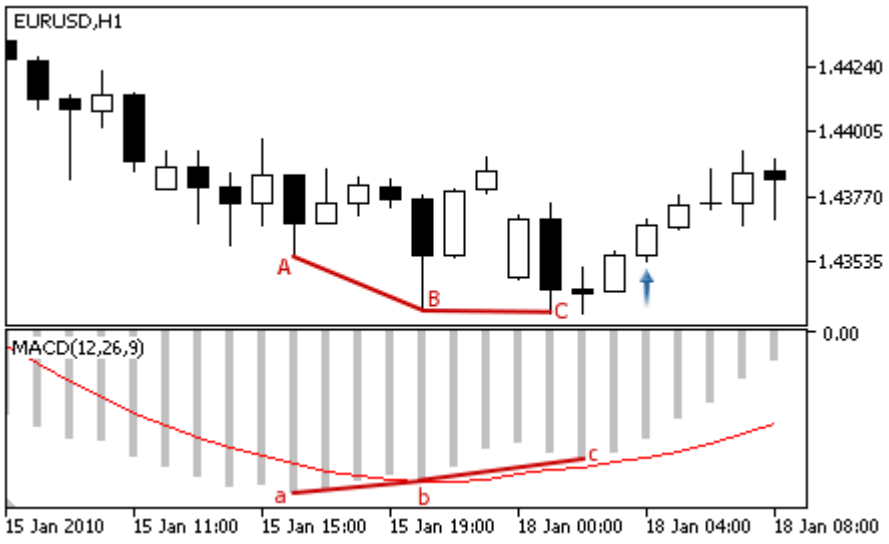
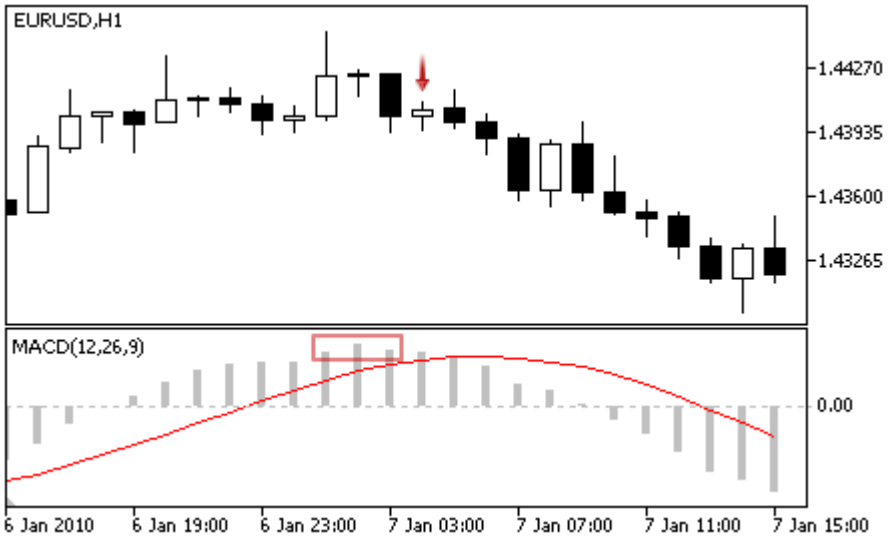


- **Дивергенция** — первая анализируемая впадина осциллятора мельче предыдущей, а соответствующая ей впадина цены глубже предыдущей.

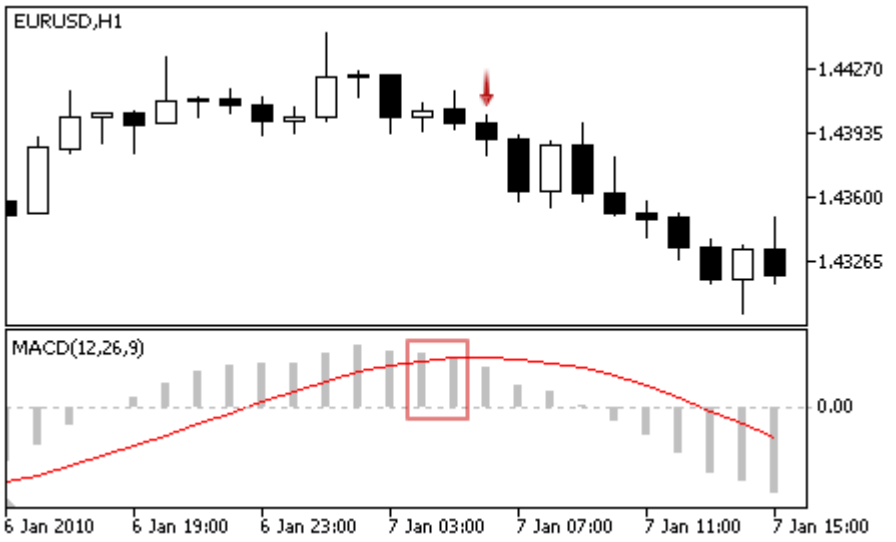
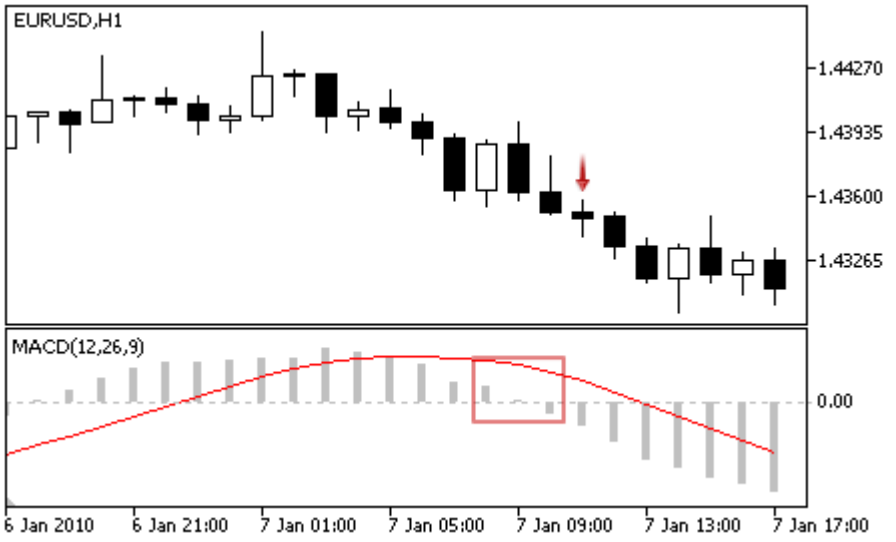


- **Двойная дивергенция** — осциллятор сформировал три последовательных впадины, каждая из которых мельче предыдущей, а цена сформировала три соответствующие им впадины, каждая из которых глубже предыдущей.

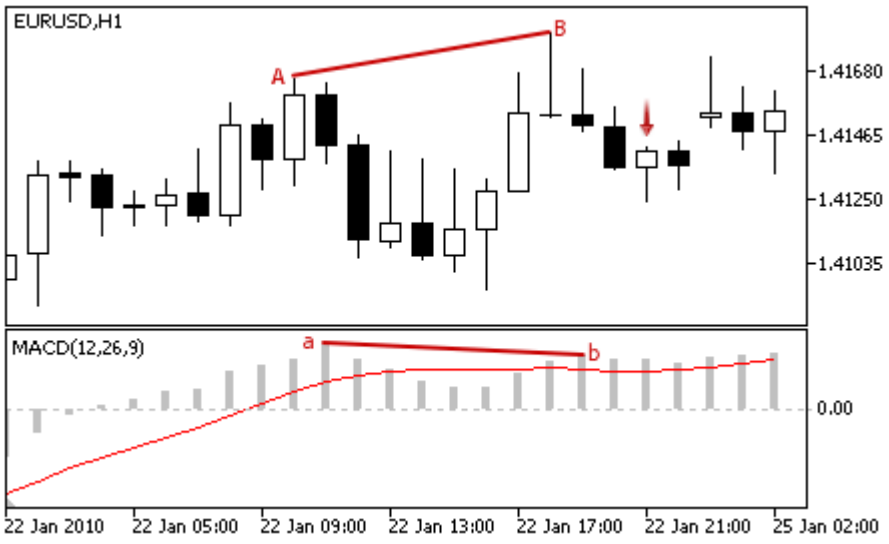
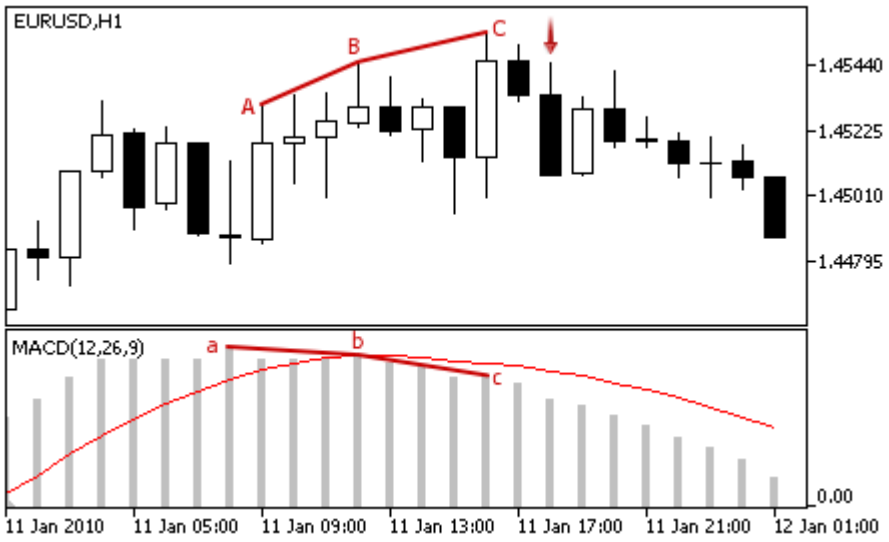
© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	
За продажу	• **Разворот** — осциллятор развернулся вниз (осциллятор падает на анализируемом баре, а на предыдущем он рос).  • **Пересечение основной и сигнальной линии** — на анализируемом баре основная линия ниже сигнальной, а на предыдущем — выше.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Пересечении нулевого уровня** — на анализируемом баре основная линия ниже нулевого уровня, а на предыдущем — выше.  • **Дивергенция** — первый анализируемый пик осциллятора ниже предыдущего, а соответствующий ему пик цены выше предыдущего.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Двойная дивергенция** — осциллятор сформировал три последовательных пика, каждый из которых ниже предыдущего, а цена сформировала три соответствующих им пика, каждый из которых выше предыдущего. 
Не против покупки	Значение осциллятора на анализируемом баре растет.
Не против продажи	Значение осциллятора на анализируемом баре падает.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodFast	Период расчета быстрой EMA.
PeriodSlow	Период расчета медленной EMA.
PeriodSignal	Период сглаживания.
Applied	Ценовой ряд, на значениях которого рассчитывается осциллятор.

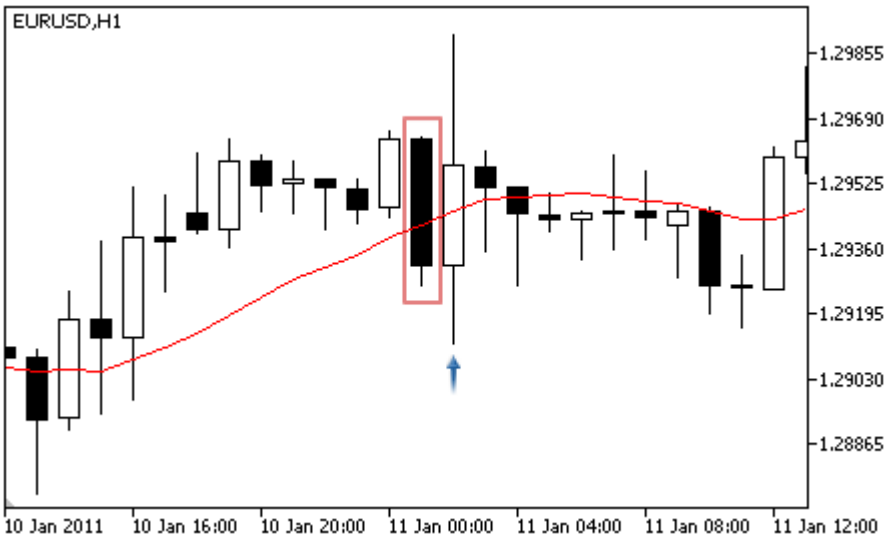
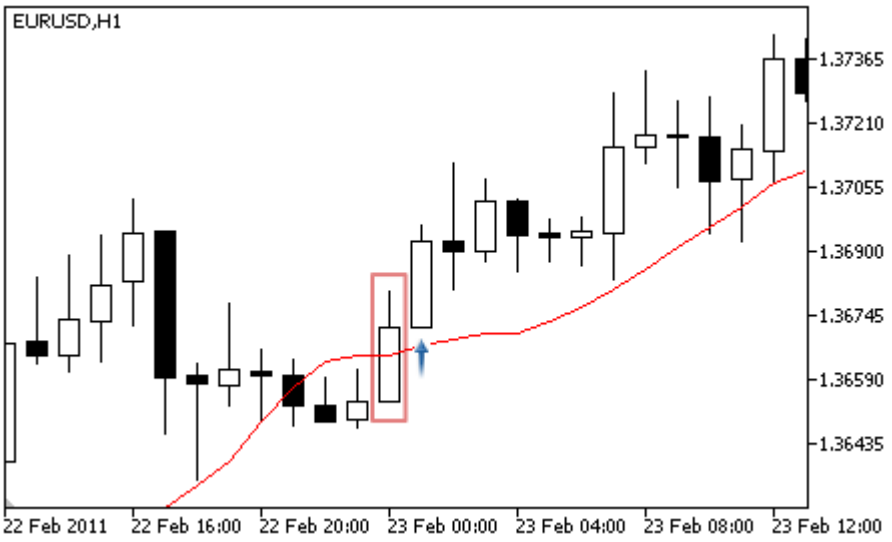
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Moving Average

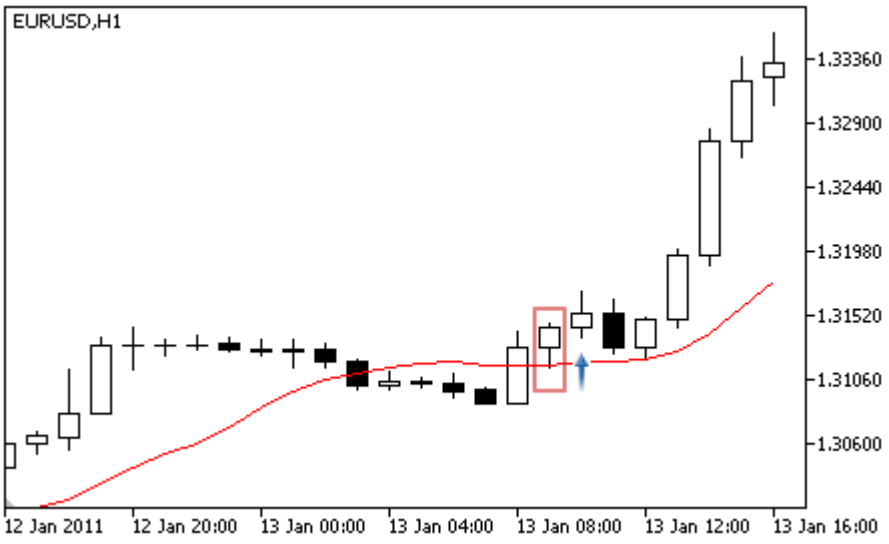
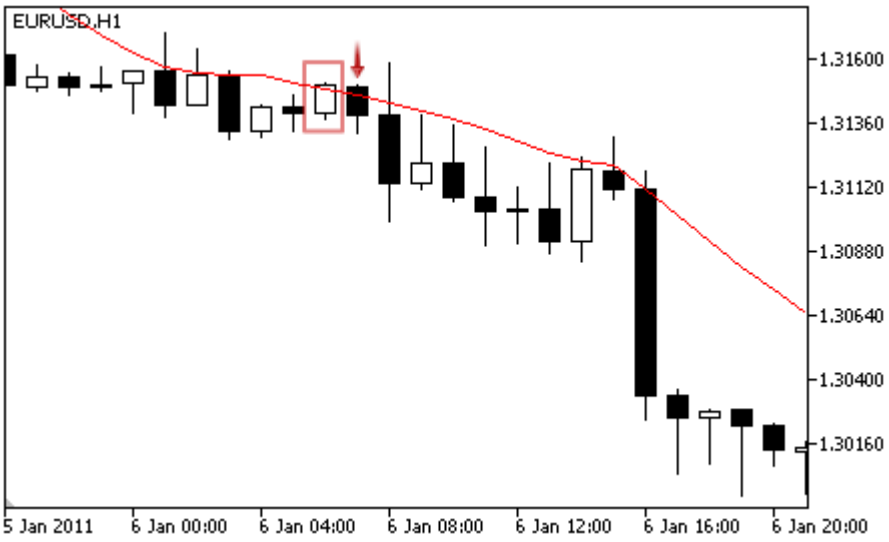
Данный модуль сигналов основан на рыночных моделях индикатора Moving Average. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

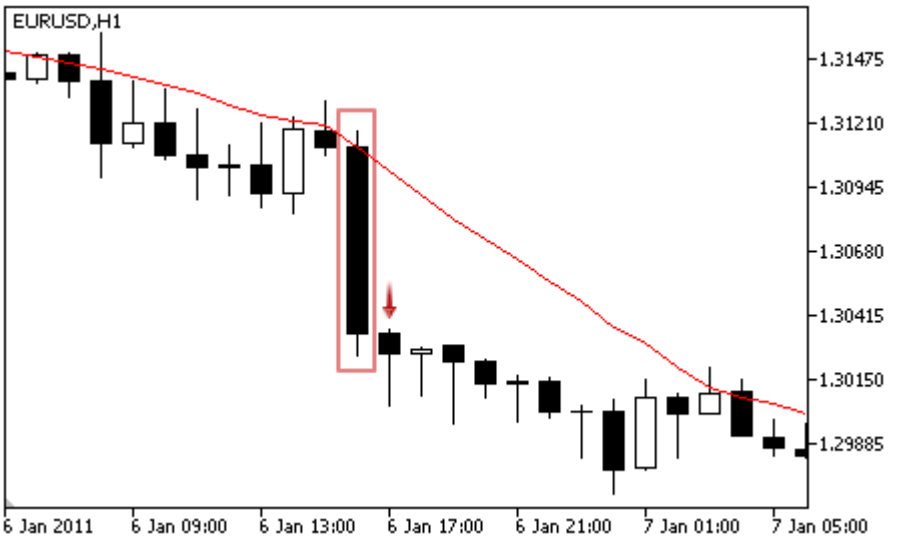
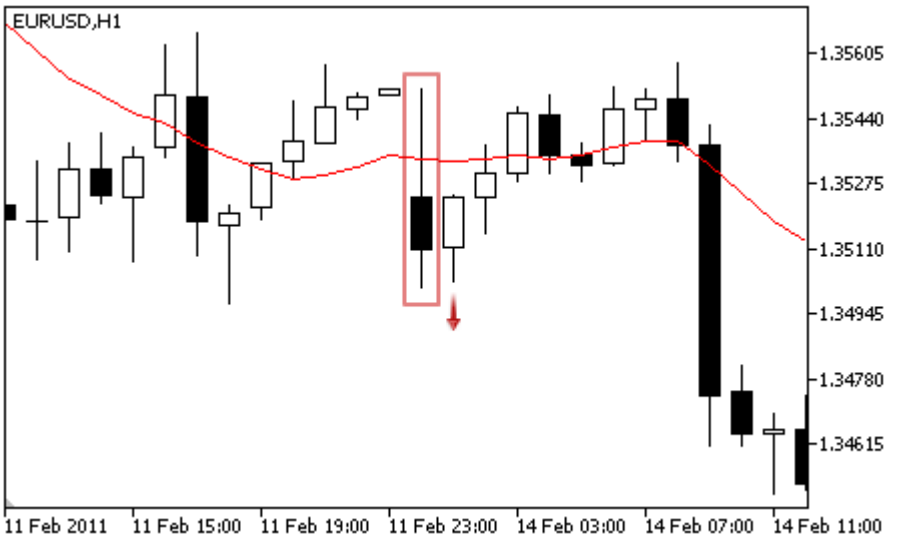
Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Несформировавшийся прокол.** Цена пересекла индикатор сверху вниз(цена Open анализируемого бара выше линии индикатора, а цена Close - ниже), но индикатор растет (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше) и индикатор растет (сильный сигнал). 

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	• **Сформировашийся прокол.** Цена пересекла индикатор нижней тенью (цены Open и Close анализируемого бара выше линии индикатора, а цена Low ниже) и индикатор растет (сигнал на отбой от линии индикатора). 
За продажу	• **Несформировавшийся прокол.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше), но индикатор падает (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор сверху вниз (цена Open анализируемого бара выше линии индикатора, а цена Close - ниже) и индикатор падает (сильный сигнал).

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Сформировашийся прокол.** Цена пересекла индикатор верхней тенью (цены Open и Close анализируемого бара ниже линии индикатора, а цена High выше) и индикатор падает (сигнал на отбой от линии индикатора). 
Не против покупки	Цена находится выше индикатора.
Не против продажи	Цена находится ниже индикатора.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodMA	Период усреднения индикатора.
Shift	Смещение индикатора по оси времени (в барах).
Method	Метод усреднения.
Applied	Ценовой ряд, на значениях которого рассчитывается индикатор.

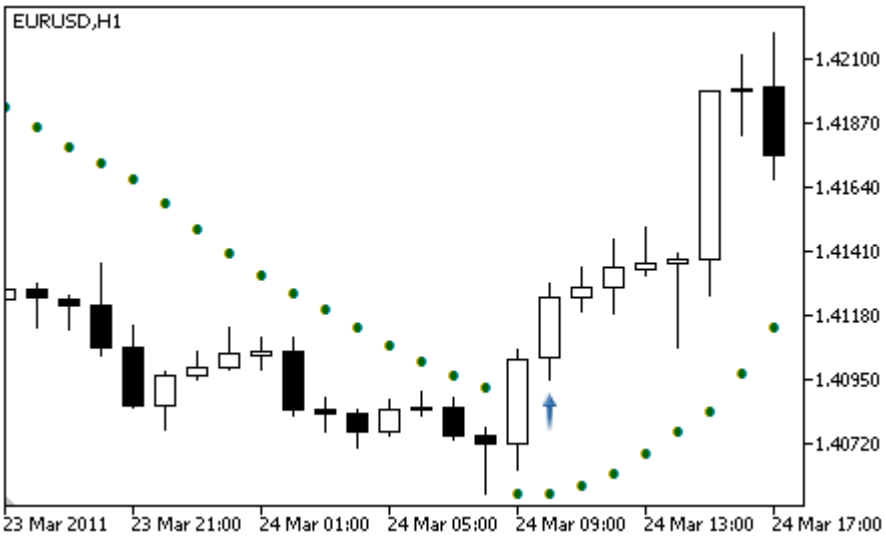
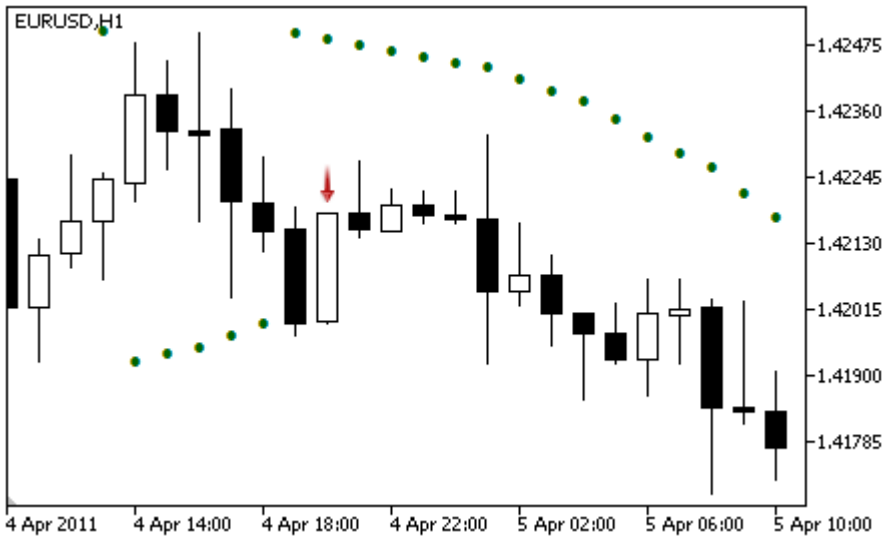
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора Parabolic SAR

Данный модуль сигналов основан на рыночных моделях индикатора Parabolic SAR. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	**Разворот** — на анализируемом баре индикатор ниже цены, а на предыдущем — выше. 
За продажу	**Разворот** — на анализируемом баре индикатор выше цены, а на предыдущем — ниже. 
Не против покупки	Цена находится выше индикатора.
Не против продажи	Цена находится ниже индикатора.

Примечание

© 2000-2025, MetaQuotes Ltd.

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
Step	Шаг изменения цены.
Maximum	Максимальный коэффициент скорости сближения индикатора с ценой.

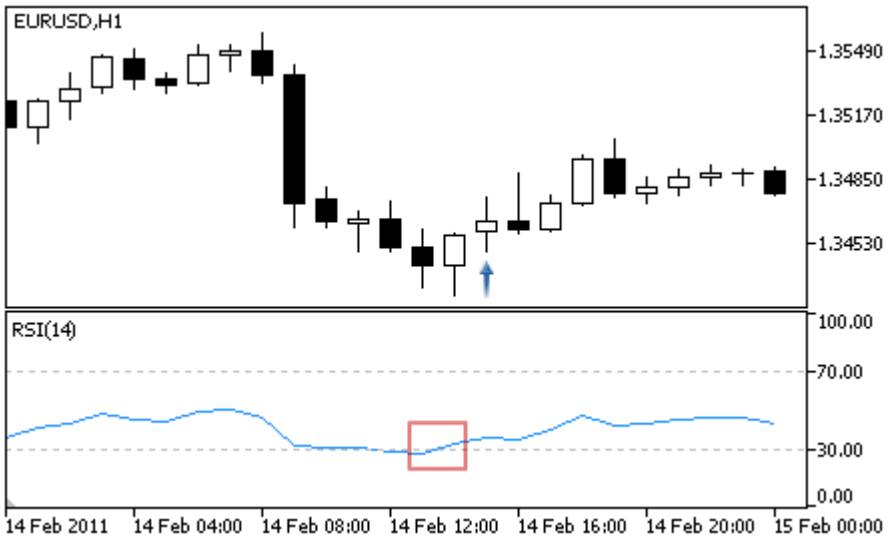
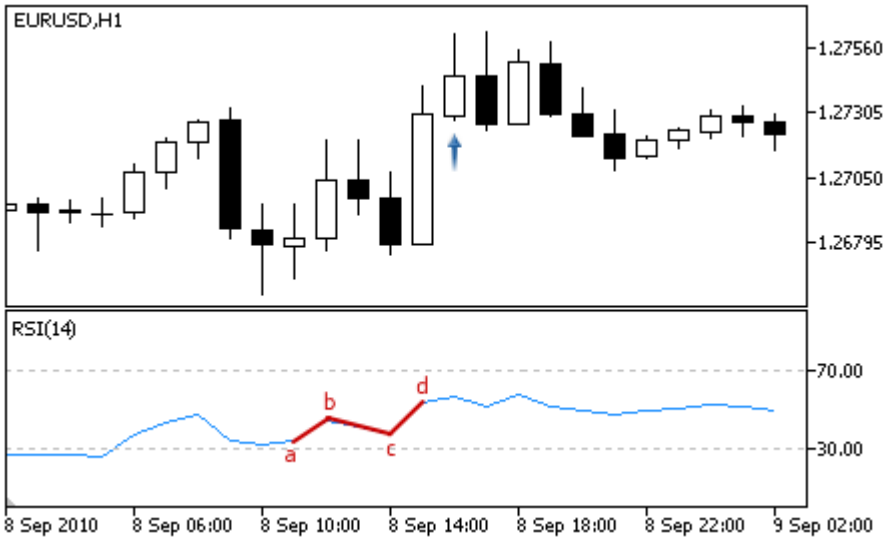
© 2000-2025, MetaQuotes Ltd.

Сигналы осциллятора Relative Strength Index

Данный модуль сигналов основан на рыночных моделях осциллятора Relative Strength Index. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

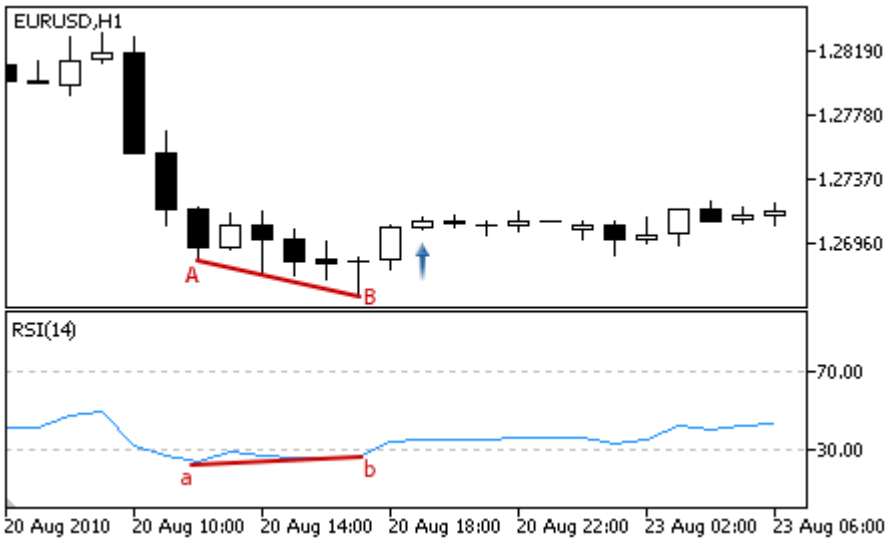
Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

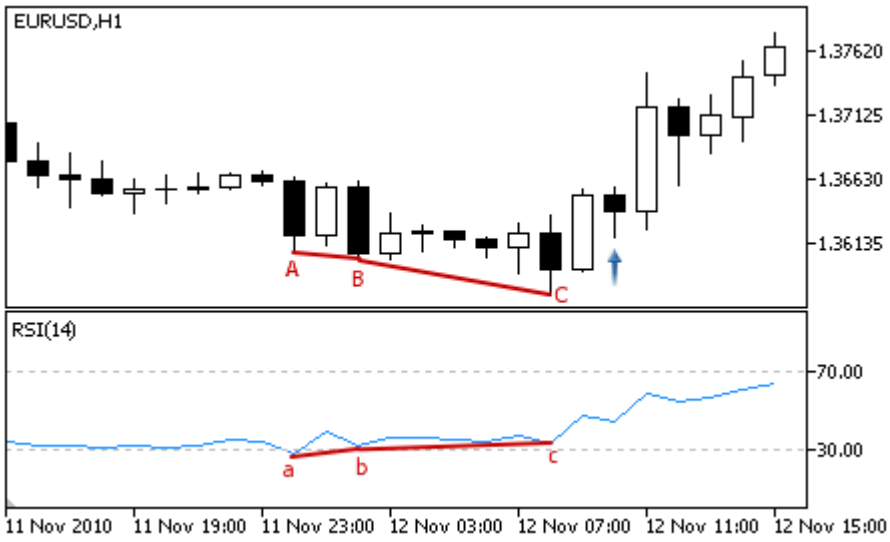
Тип сигнала	Описание условий
За покупку	• **Разворот за уровнем перепроданности** — осциллятор развернулся вверх и его значение на анализируемом баре находится за уровнем перепроданности (по умолчанию 30).  • **Неудавшийся размах** — на анализируемом баре осциллятор поднялся выше предыдущего пика.  • **Дивергенция** — первая анализируемая впадина осциллятора мельче предыдущей, а соответствующая ей впадина цены глубже предыдущей.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий

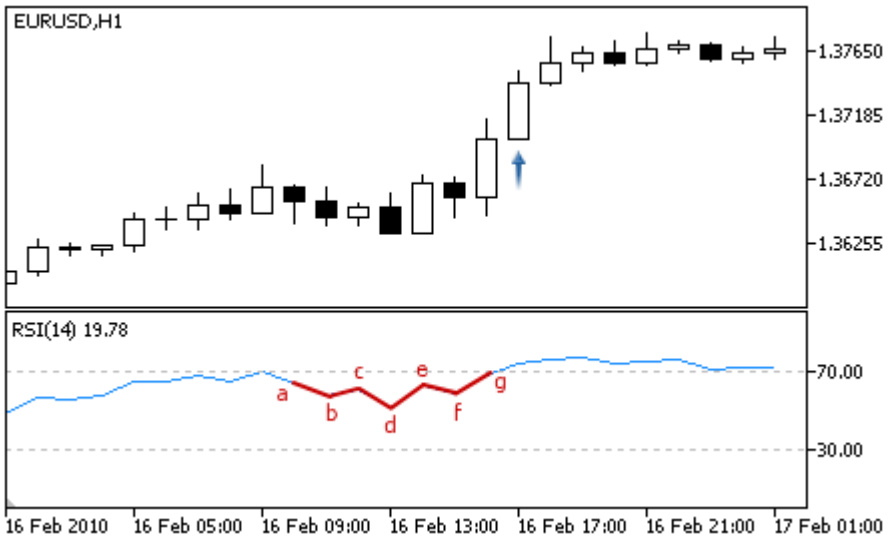
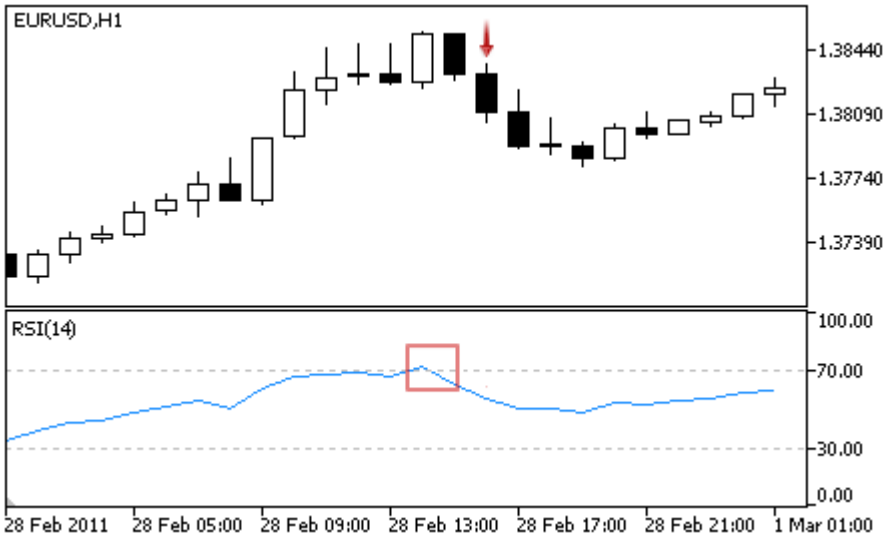


- **Двойная дивергенция** — осциллятор сформировал три последовательных впадины, каждая из которых мельче предыдущей, а цена сформировала три соответствующие им впадины, каждая из которых глубже предыдущей.



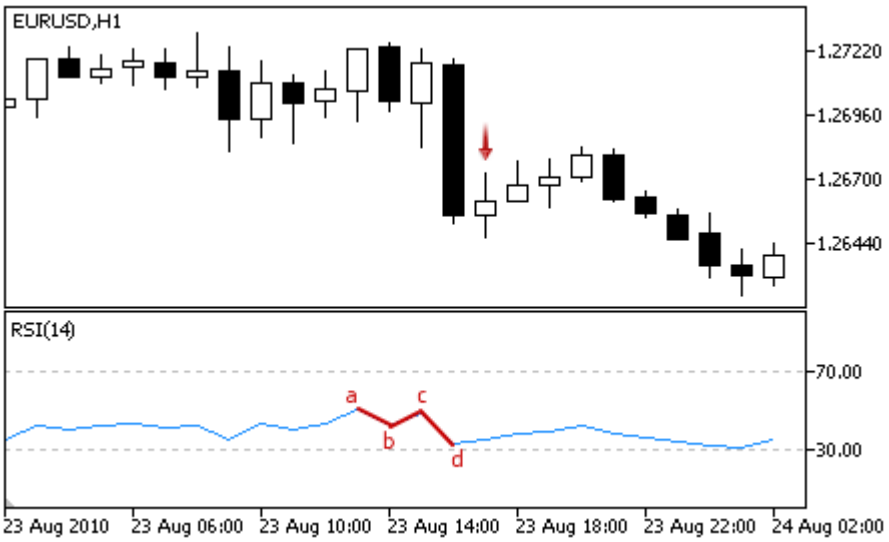
- **Голова/Плечи** — осциллятор сформировал три последовательных впадины, средняя из которых глубже двух остальных.

© 2000-2025, MetaQuotes Ltd.

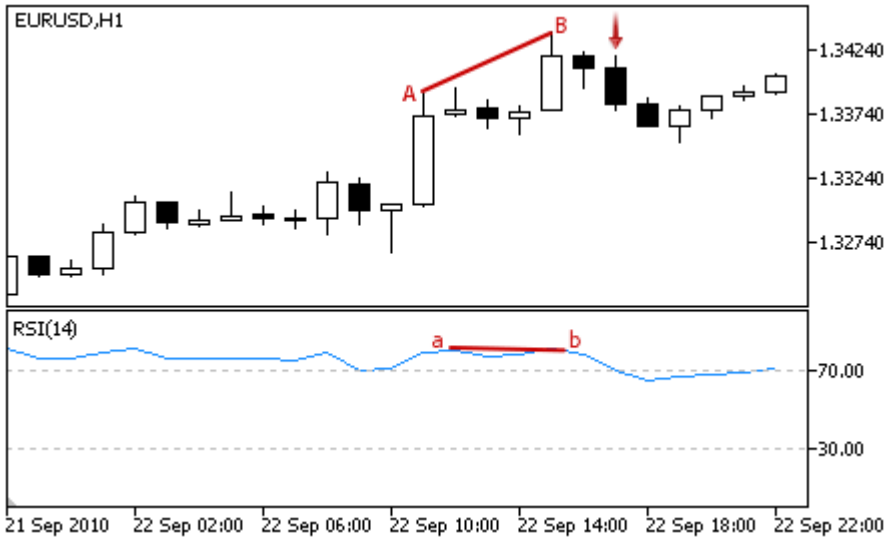
Тип сигнала	Описание условий
	
За продажу	• **Разворот за уровнем перекупленности** — осциллятор развернулся вниз и его значение на анализируемом баре находится за уровнем перекупленности (по умолчанию 70).  • **Неудавшийся размах** — на анализируемом баре осциллятор опустился ниже предыдущей впадины.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий

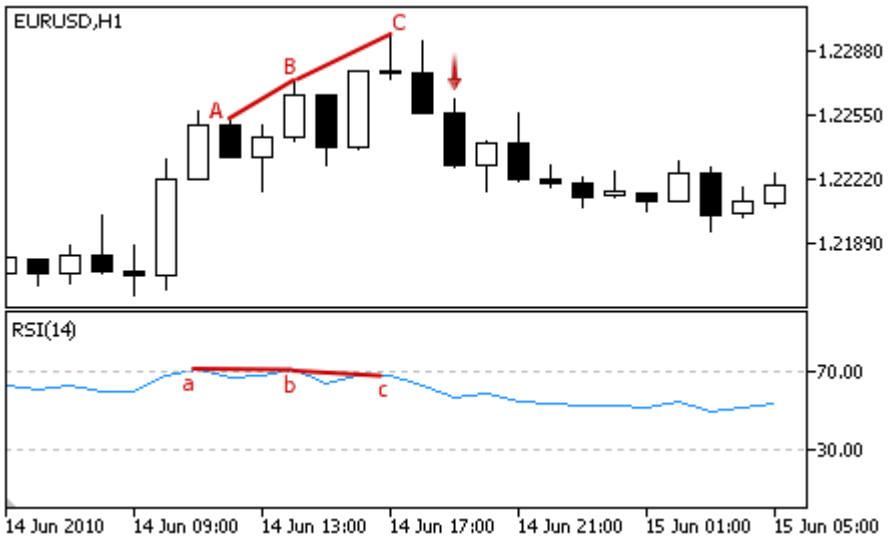
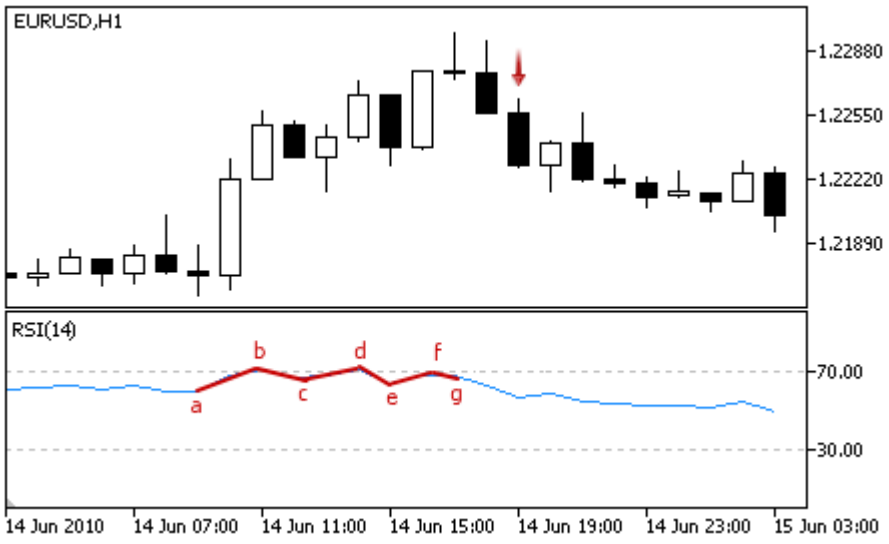


- **Дивергенция** — первый анализируемый пик осциллятора ниже предыдущего, а соответствующий ему пик цены выше предыдущего.



- **Двойная дивергенция** — осциллятор сформировал три последовательных пика, каждый из которых ниже предыдущего, а цена сформировала три соответствующих им пика, каждый из которых выше предыдущего.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Голова/Плечи** — осциллятор сформировал три последовательных пика, средний из которых выше двух остальных. 
Не против покупки	Значение осциллятора на анализируемом баре растет.
Не против продажи	Значение осциллятора на анализируемом баре падает.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodRSI	Период расчета осциллятора.
Applied	Ценовой ряд, на значениях которого рассчитывается осциллятор.

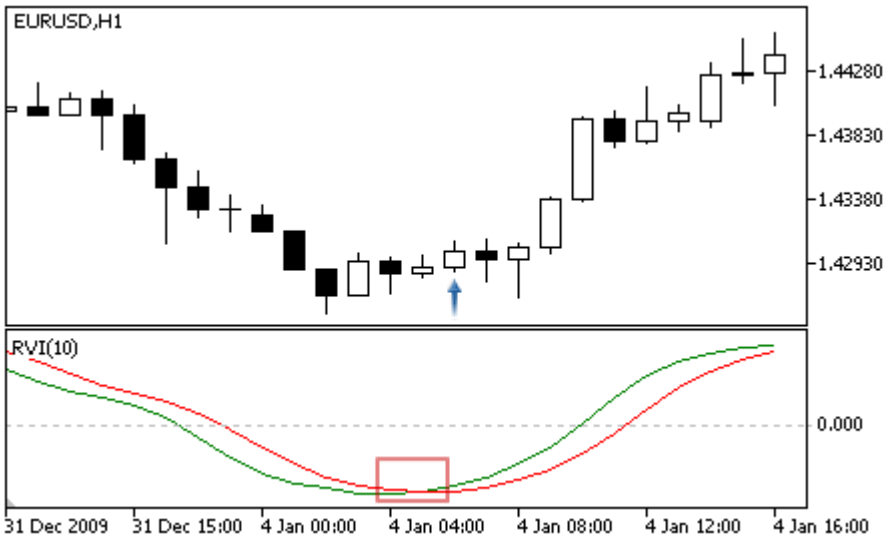
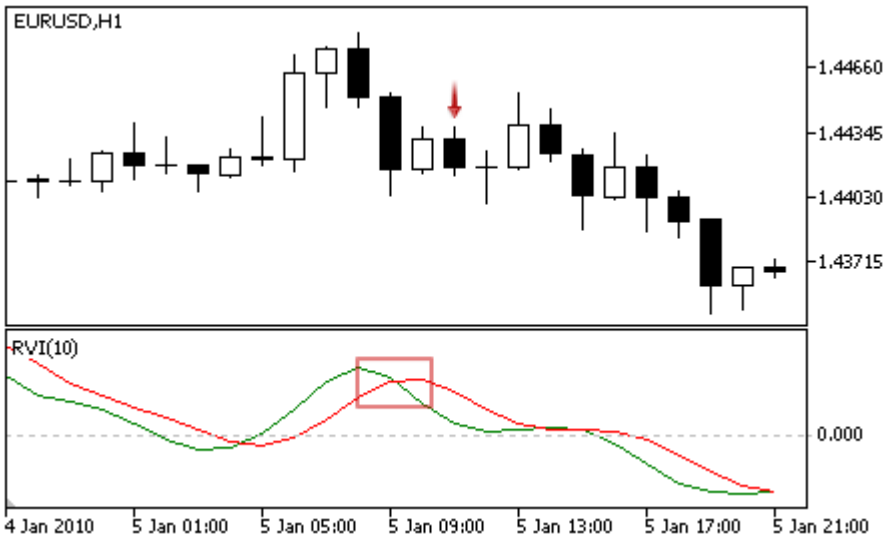
© 2000-2025, MetaQuotes Ltd.

Сигналы осциллятора Relative Vigor Index

Данный модуль сигналов основан на рыночных моделях осциллятора Relative Vigor Index. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	**Пересечение основной и сигнальной линии** — на анализируемом баре основная линия выше сигнальной, а на предыдущем — ниже. 
За продажу	**Пересечение основной и сигнальной линии** — на анализируемом баре основная линия ниже сигнальной, а на предыдущем — выше. 
Не против покупки	Значение осциллятора на анализируемом баре растет.
Не против продажи	Значение осциллятора на анализируемом баре падает.

Примечание

© 2000-2025, MetaQuotes Ltd.

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodRVI	Период расчета осциллятора.

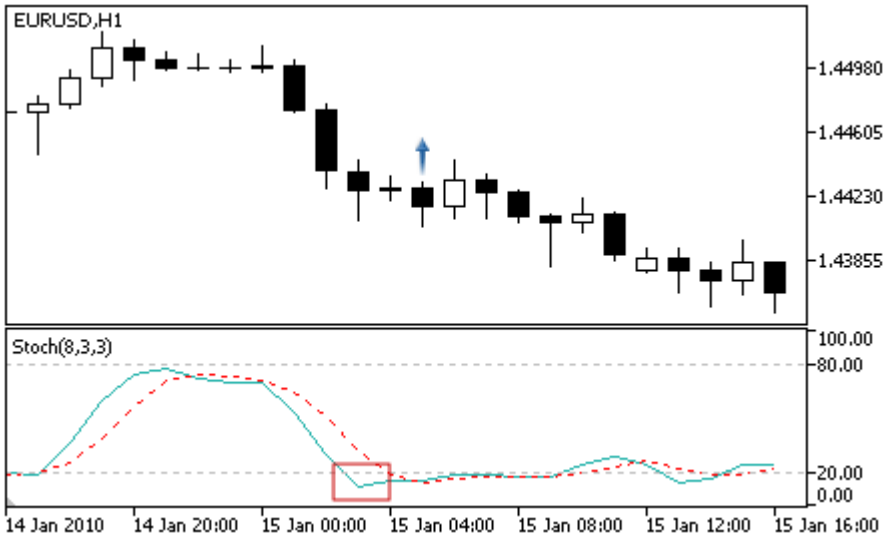
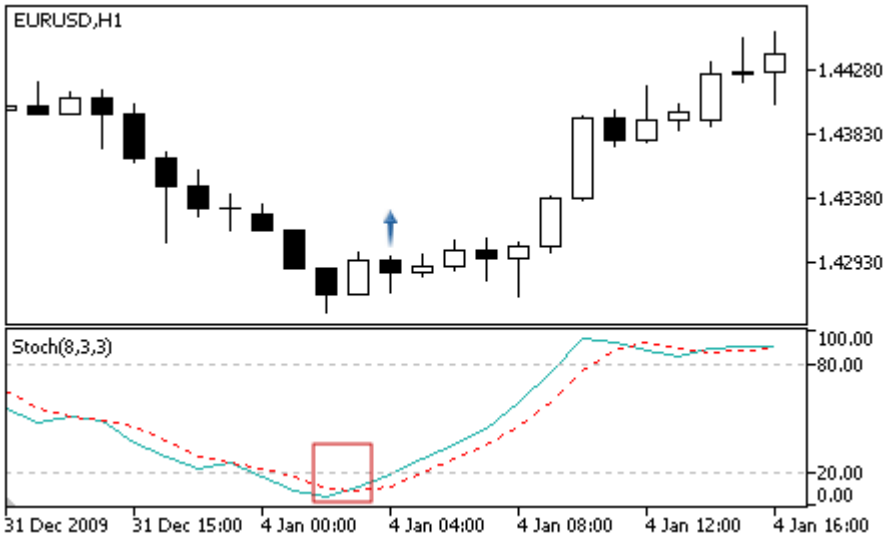
© 2000-2025, MetaQuotes Ltd.

Сигналы осциллятора Stochastic

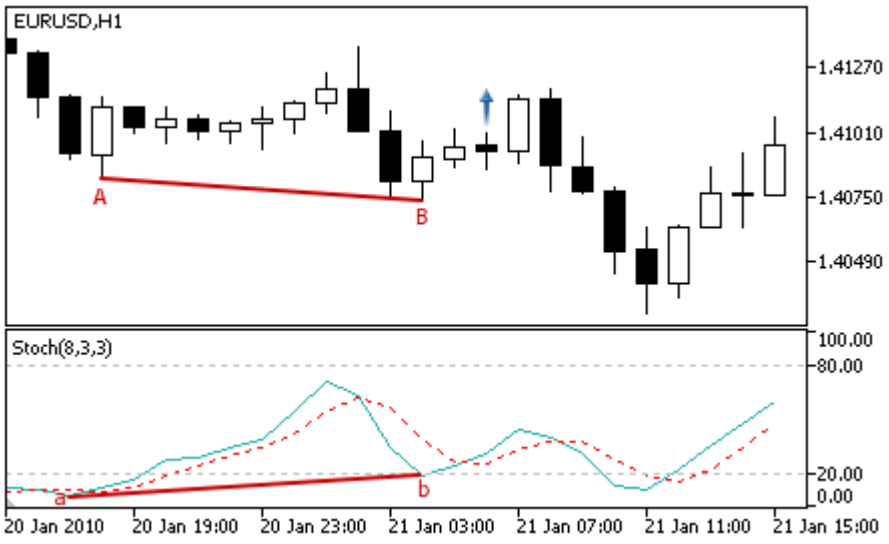
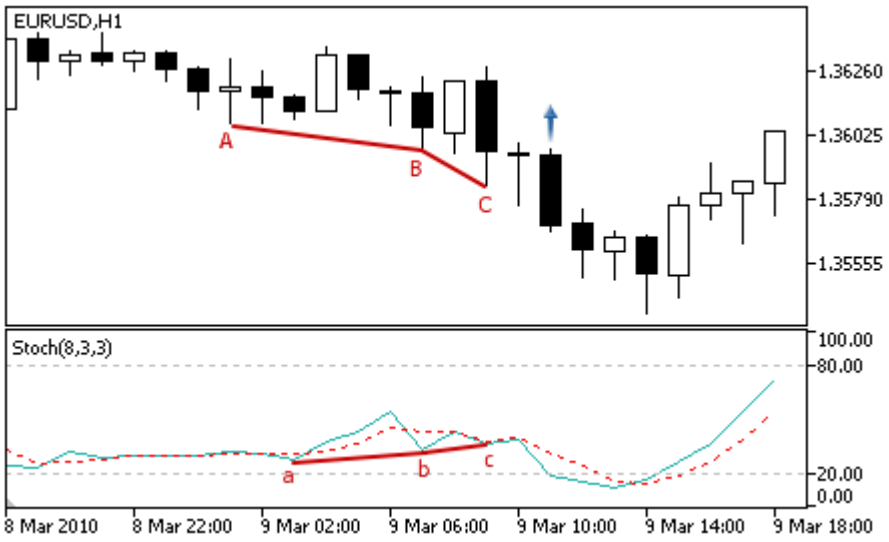
Данный модуль сигналов основан на рыночных моделях осциллятора Stochastic. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

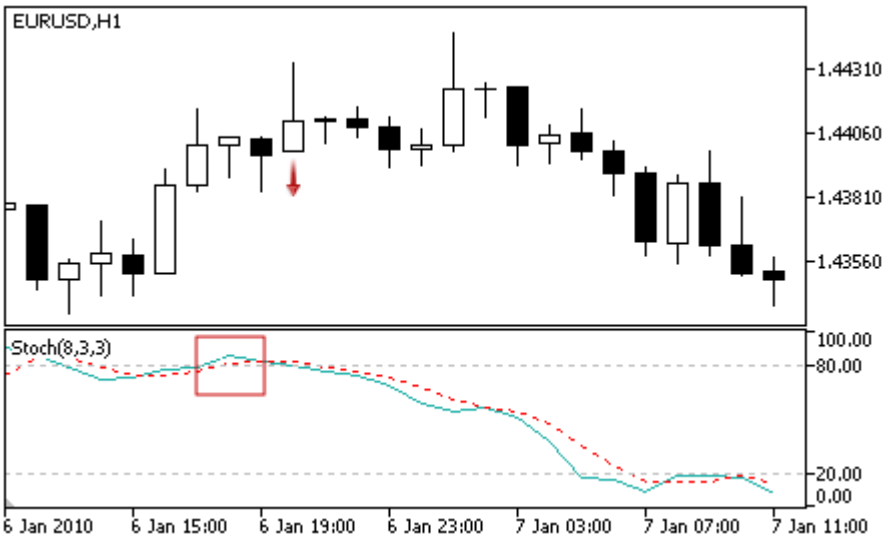
Тип сигнала	Описание условий
За покупку	• **Разворот** — осциллятор развернулся вверх (осциллятор растет на анализируемом баре, а на предыдущем он падал).  • **Пересечение основной и сигнальной линии** — на анализируемом баре основная линия выше сигнальной, а на предыдущем — ниже.  • **Дивергенция** — первая анализируемая впадина осциллятора мельче предыдущей, а соответствующая ей впадина цены глубже предыдущей.

© 2000-2025, MetaQuotes Ltd.

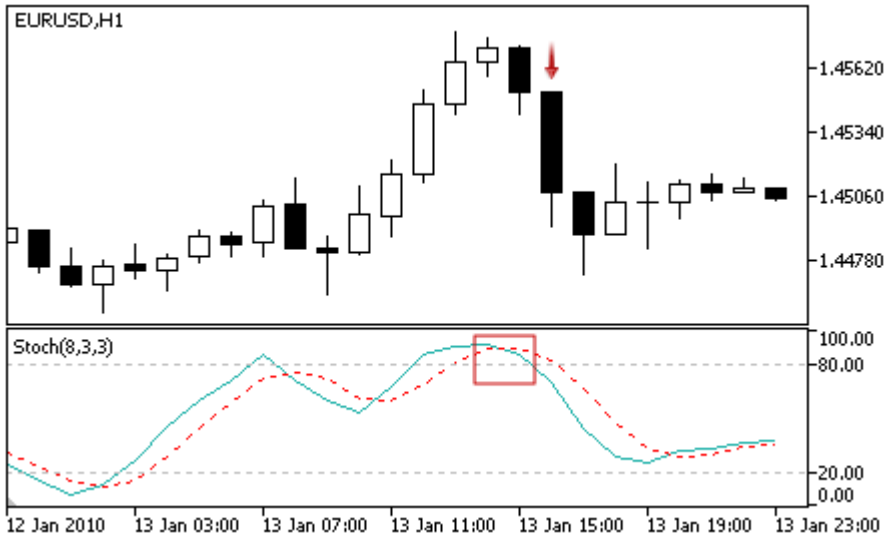
Тип сигнала	Описание условий
	 • **Двойная дивергенция** — осциллятор сформировал три последовательных впадины, каждая из которых мельче предыдущей, а цена сформировала три соответствующие им впадины, каждая из которых глубже предыдущей. 
За продажу	• **Разворот** — осциллятор развернулся вниз (осциллятор падает на анализируемом баре, а на предыдущем он рос).

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий

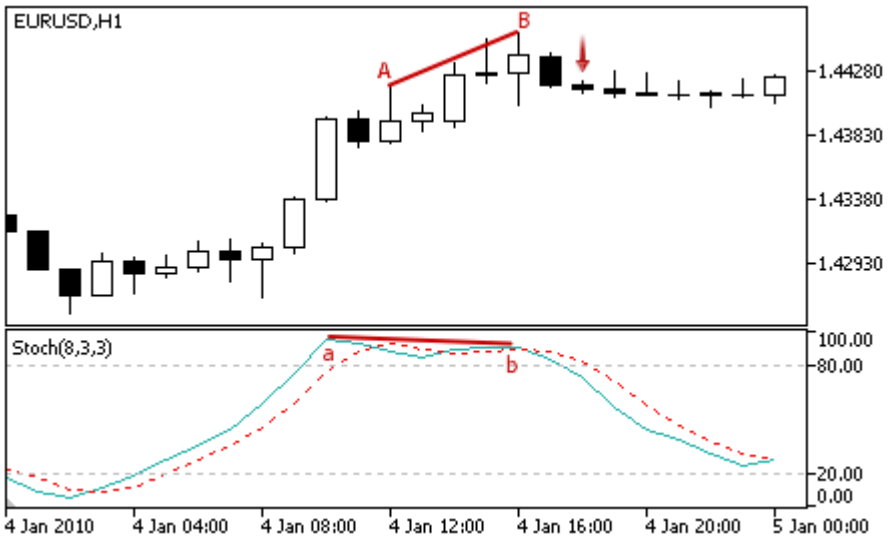
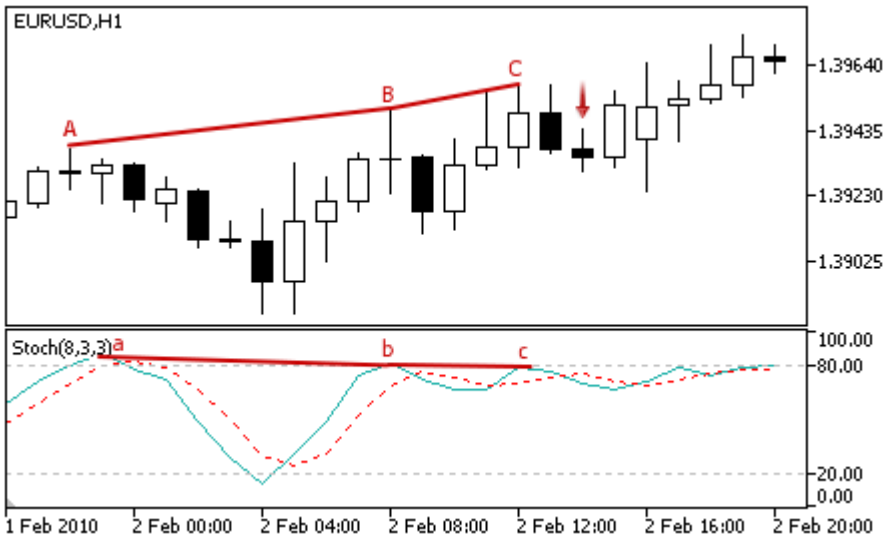


- **Пересечение основной и сигнальной линии** — на анализируемом баре основная линия ниже сигнальной, а на предыдущем — выше.



- **Дивергенция** — первый анализируемый пик осциллятора ниже предыдущего, а соответствующий ему пик цены выше предыдущего.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Двойная дивергенция** — осциллятор сформировал три последовательных пика, каждый из которых ниже предыдущего, а цена сформировала три соответствующих им пика, каждый из которых выше предыдущего. 
Не против покупки	Значение осциллятора на анализируемом баре растет.
Не против продажи	Значение осциллятора на анализируемом баре падает.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodK	Период расчета главной линии осциллятора.
PeriodD	Период усреднения главной линии осциллятора.
PeriodSlow	Период замедления.
Applied	Ценовой ряд, на значениях которого рассчитывается осциллятор.

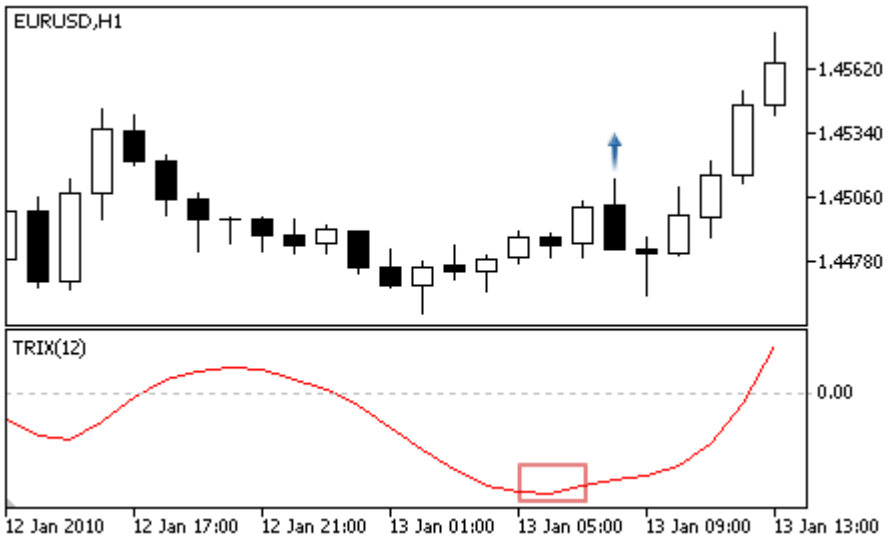
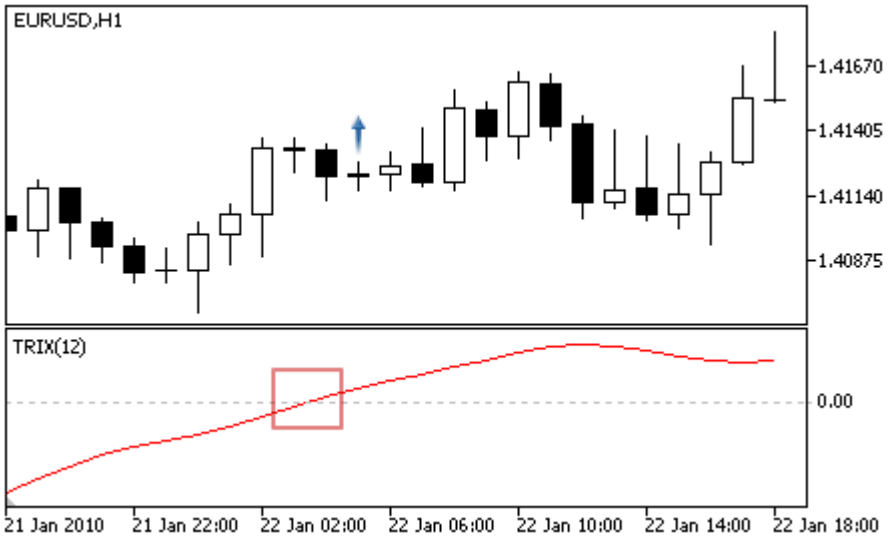
© 2000-2025, MetaQuotes Ltd.

Сигналы осциллятора Triple Exponential Average

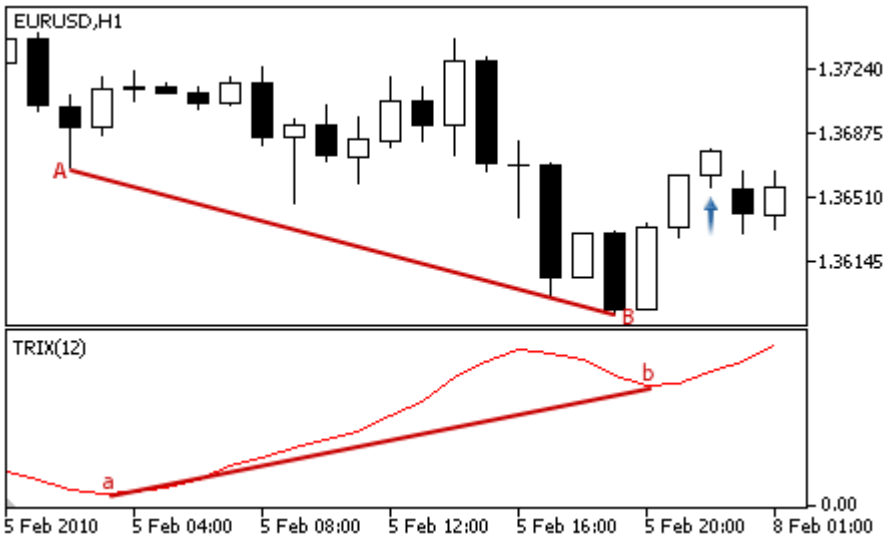
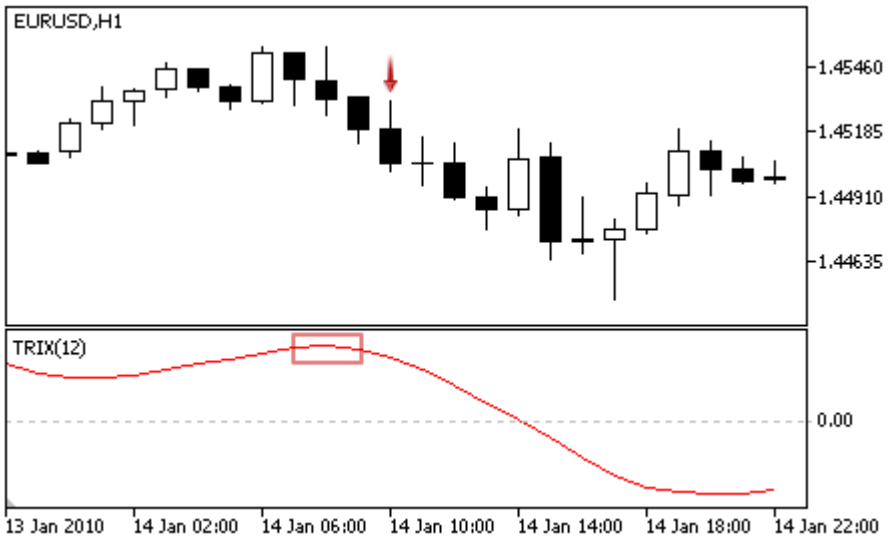
Данный модуль сигналов основан на рыночных моделях осциллятора Triple Exponential Average. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

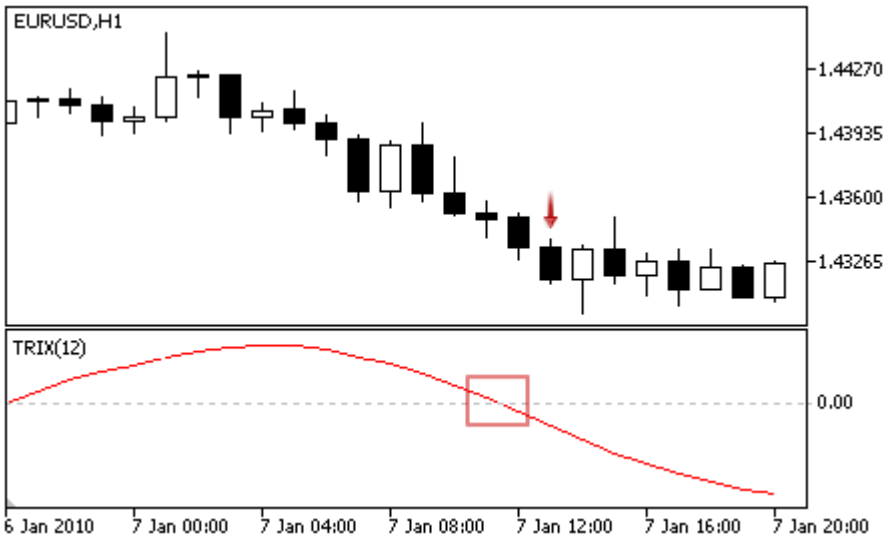
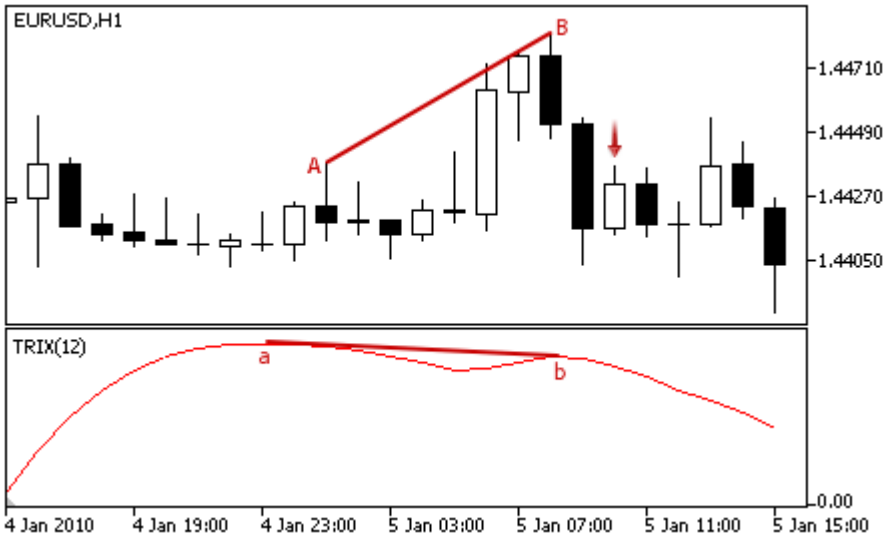
Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Разворот** — осциллятор развернулся вверх (осциллятор растет на анализируемом баре, а на предыдущем он падал).  • **Пересечении нулевого уровня** — на анализируемом баре основная линия выше нулевого уровня, а на предыдущем — ниже.  • **Дивергенция** — первая анализируемая впадина осциллятора мельче предыдущей, а соответствующая ей впадина цены глубже предыдущей.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	
За продажу	• **Разворот** — осциллятор развернулся вниз (осциллятор падает на анализируемом баре, а на предыдущем он рос).  • **Пересечении нулевого уровня** — на анализируемом баре основная линия ниже нулевого уровня, а на предыдущем — выше.

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Дивергенция** — первый анализируемый пик осциллятора ниже предыдущего, а соответствующий ему пик цены выше предыдущего. 
Не против покупки	Значение осциллятора на анализируемом баре растет.
Не против продажи	Значение осциллятора на анализируемом баре падает.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodTriX	Период расчета осциллятора.
Applied	Ценовой ряд, на значениях которого рассчитывается осциллятор.

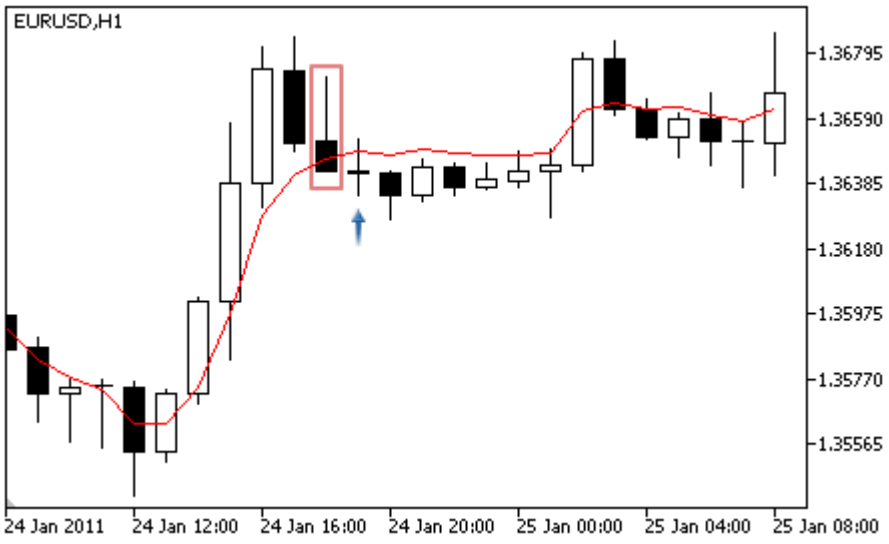
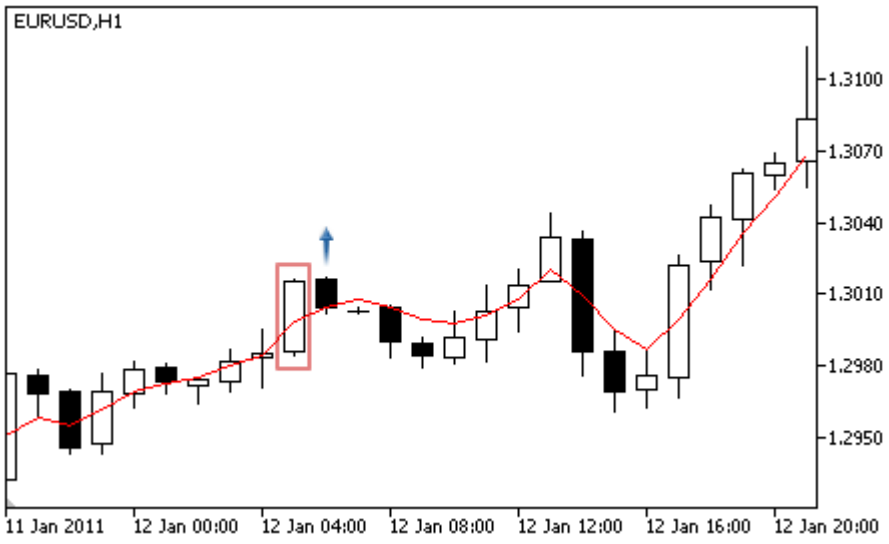
© 2000-2025, MetaQuotes Ltd.

Сигналы индикатора

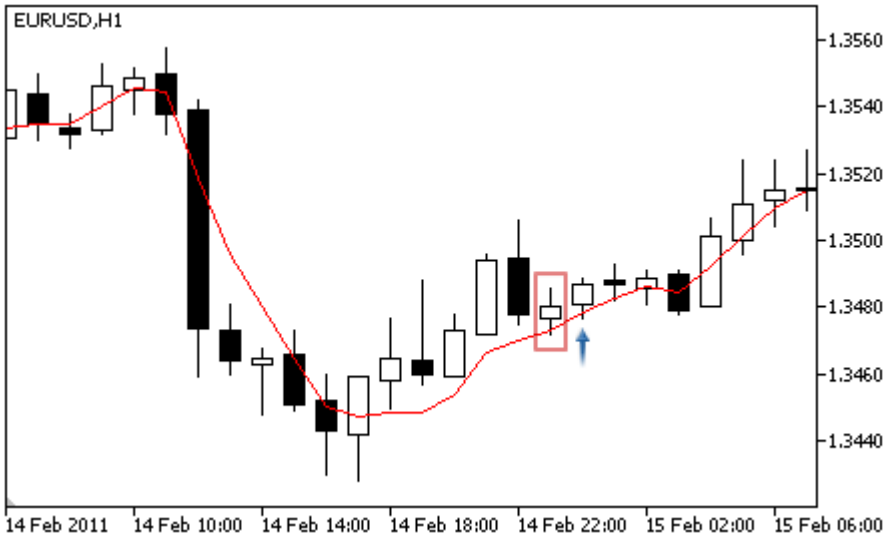
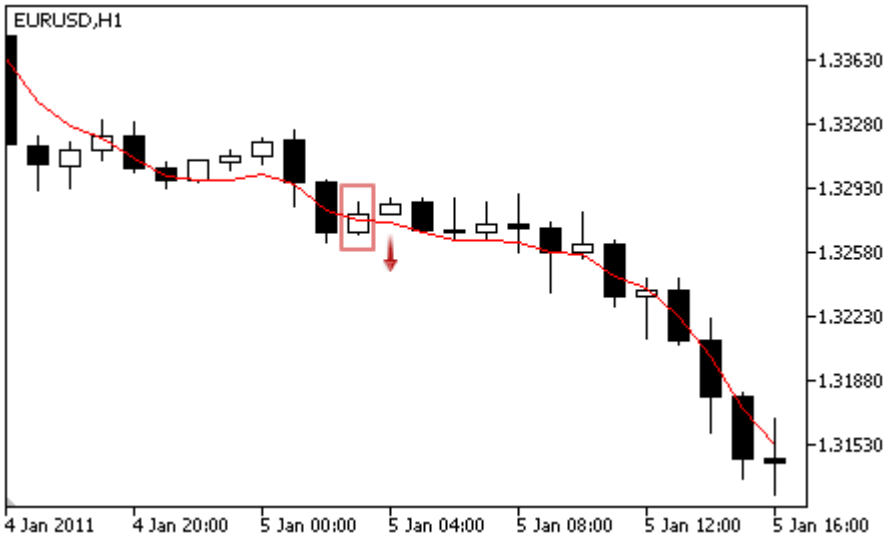
Данный модуль сигналов основан на рыночных моделях индикатора Triple Exponential Moving Average. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

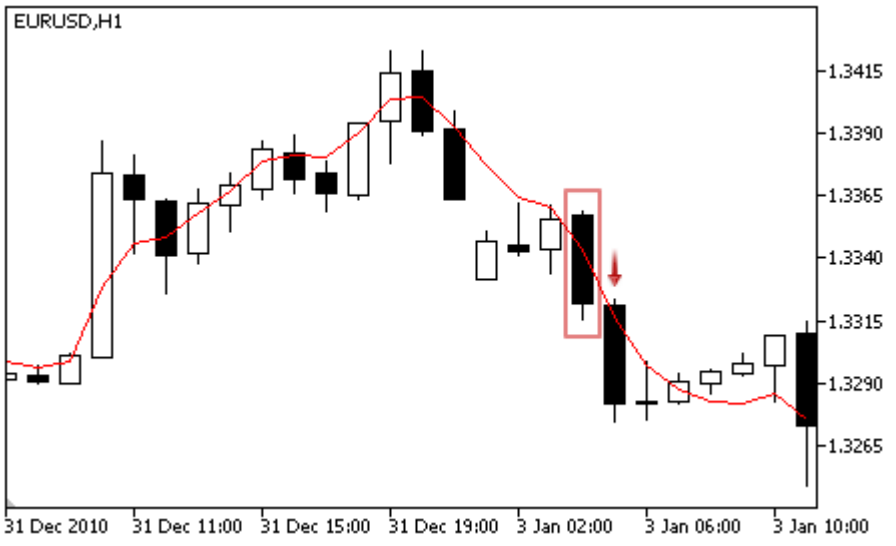
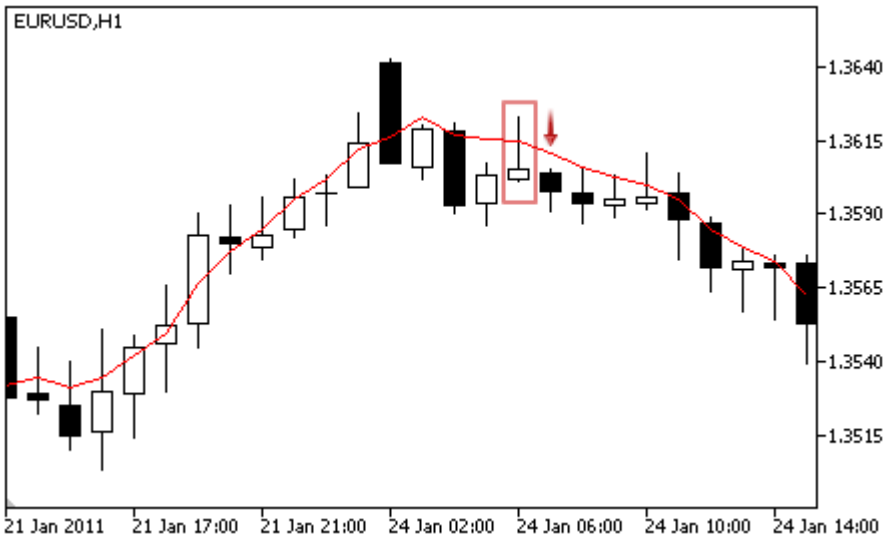
Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Несформировавшийся прокол.** Цена пересекла индикатор сверху вниз(цена Open анализируемого бара выше линии индикатора, а цена Close - ниже), но индикатор растет (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше) и индикатор растет (сильный сигнал). 

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	• **Сформировашийся прокол.** Цена пересекла индикатор нижней тенью (цены Open и Close анализируемого бара выше линии индикатора, а цена Low ниже) и индикатор растет (сигнал на отбой от линии индикатора). 
За продажу	• **Несформировавшийся прокол.** Цена пересекла индикатор снизу вверх (цена Open анализируемого бара ниже линии индикатора, а цена Close - выше), но индикатор падает (слабый сигнал на отбой от линии индикатора).  • **Пересечение скользящей средней.** Цена пересекла индикатор сверху вниз (цена Open анализируемого бара выше линии индикатора, а цена Close - ниже) и индикатор падает (сильный сигнал).

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	 • **Сформировашийся прокол.** Цена пересекла индикатор верхней тенью (цены Open и Close анализируемого бара ниже линии индикатора, а цена High выше) и индикатор падает (сигнал на отбой от линии индикатора). 
Не против покупки	Цена находится выше индикатора.
Не против продажи	Цена находится ниже индикатора.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Настраиваемые параметры

Данный модуль обладает следующими настраиваемыми параметрами:

© 2000-2025, MetaQuotes Ltd.

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodMA	Период усреднения индикатора.
Shift	Смещение индикатора по оси времени (в барах).
Method	Метод усреднения.
Applied	Ценовой ряд, на значениях которого рассчитывается индикатор.

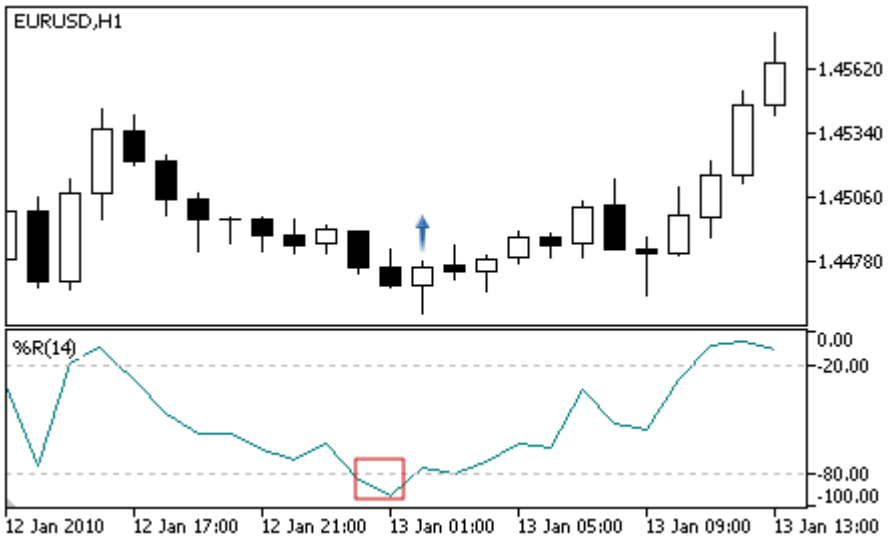
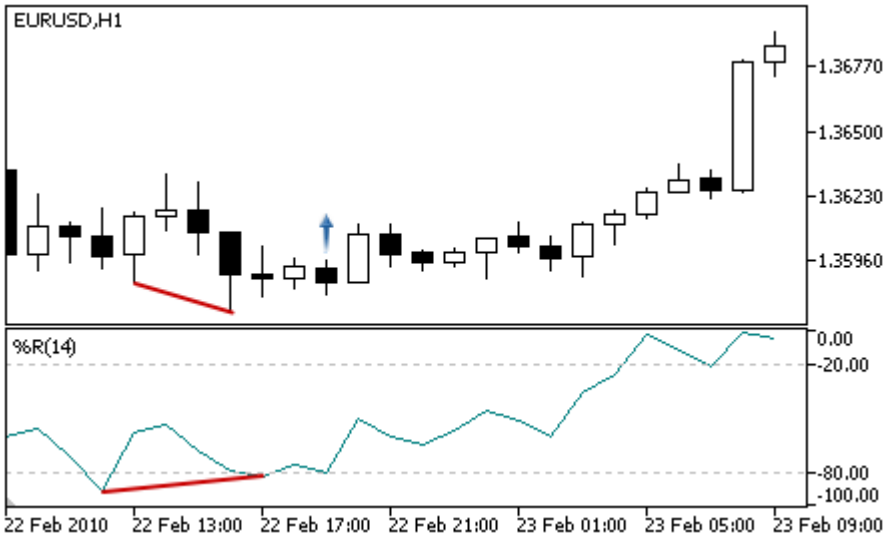
© 2000-2025, MetaQuotes Ltd.

Сигналы осциллятора Williams Percent Range

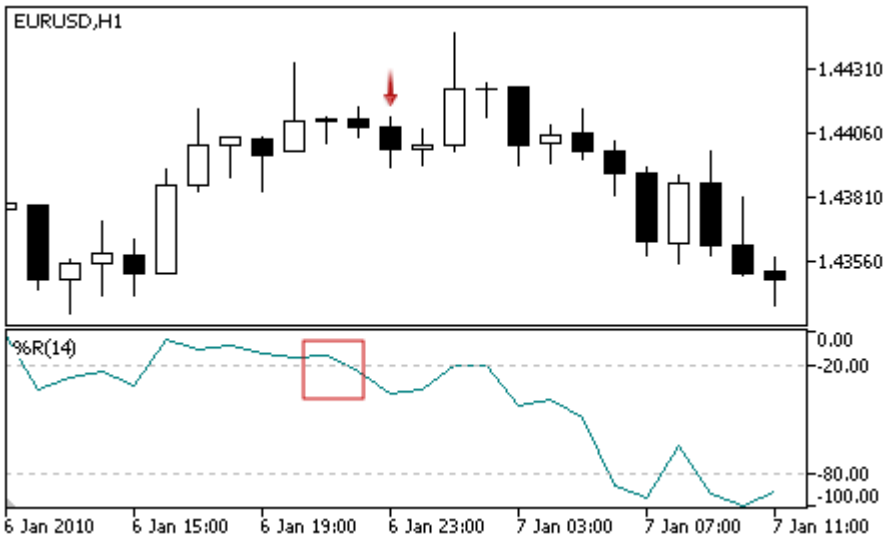
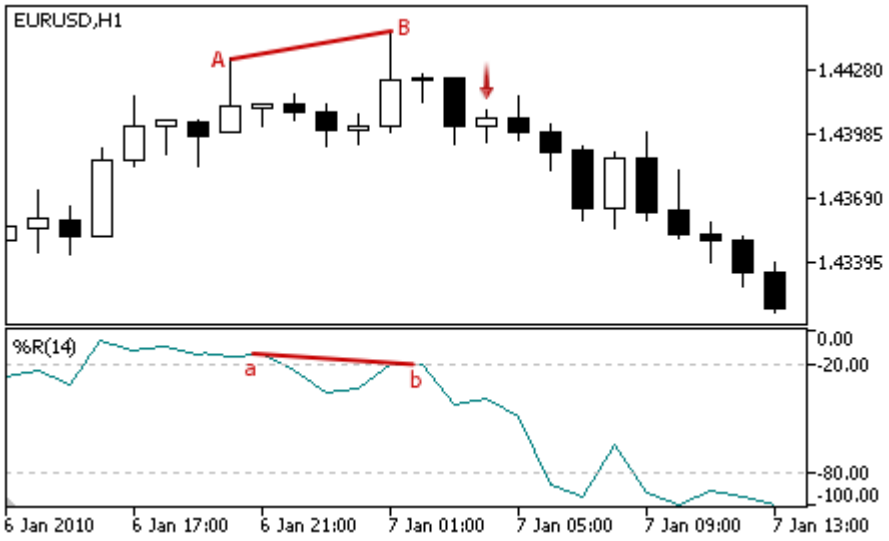
Данный модуль сигналов основан на рыночных моделях осциллятора Williams Percent Range. Механизм принятия торговых решений на основе сигналов модулей описан в отдельном разделе.

Условия генерации сигналов

Ниже приведено описание условий, при которых модуль передает советнику тот или иной сигнал.

Тип сигнала	Описание условий
За покупку	• **Разворот за уровнем перепроданности** — осциллятор развернулся вверх и его значение на анализируемом баре находится за уровнем перепроданности (по умолчанию -80).  • **Дивергенция** — первая анализируемая впадина осциллятора мельче предыдущей, а соответствующая ей впадина цены глубже предыдущей. 
За продажу	• **Разворот за уровнем перекупленности** — осциллятор развернулся вниз и его значение на анализируемом баре находится за уровнем

© 2000-2025, MetaQuotes Ltd.

Тип сигнала	Описание условий
	перекупленности (по умолчанию -20).  • **Дивергенция** — первый анализируемый пик осциллятора ниже предыдущего, а соответствующий ему пик цены выше предыдущего. 
Не против покупки	Значение осциллятора на анализируемом баре растет.
Не против продажи	Значение осциллятора на анализируемом баре падает.

Примечание

В зависимости от режима работы эксперта ("Каждый тик" или "По ценам открытия") анализируемый бар — это либо текущий бар (с индексом 0), либо последний сформировавшийся бар (с индексом 1).

Следует учитывать, что осциллятор Williams Percent Range имеет перевернутую шкалу. Его максимальное значение -100, а минимальное 0.

Настраиваемые параметры

© 2000-2025, MetaQuotes Ltd.

Данный модуль обладает следующими настраиваемыми параметрами:

Параметр	Описание
Weight	Вес сигнала модуля в интервале от 0 до 1.
PeriodWPR	Период расчета осциллятора.

© 2000-2025, MetaQuotes Ltd.

Набор готовых реализаций алгоритмов сопровождения открытых позиций

Этот раздел содержит технические детали работы с классами готовых реализаций алгоритмов сопровождения открытых позиций и описание соответствующих компонентов стандартной библиотеки MQL5.

Использование классов готовых реализаций алгоритмов сопровождения открытых позиций позволит сэкономить время при разработке (а особенно, проверке) торговых стратегий.

Стандартная библиотека MQL5 (в части набора готовых реализаций алгоритмов сопровождения открытых позиций) размещается в рабочем каталоге терминала в папке Include\Expert\Trailing.

Класс	Описание
CTrailingFixedPips	Класс реализации алгоритма сопровождения открытых позиций на фиксированном "расстоянии"
CTrailingMA	Класс реализации алгоритма сопровождения открытых позиций по значениям скользящей средней
CTrailingNone	Класс-заглушка алгоритмов сопровождения открытых позиций
CTrailingPSAR	Класс реализации алгоритма сопровождения открытых позиций по значениям индикатора Parabolic SAR

© 2000-2025, MetaQuotes Ltd.

Класс CTrailingFixedPips

Класс CTrailingFixedPips является классом реализации алгоритма сопровождения открытых позиций на фиксированном "расстоянии" (в пунктах).

Класс CTrailingFixedPips реализует следующий алгоритм сопровождения открытых позиций: Если уровень Stop Loss равен нулю, считается, что условие модификации ордера Stop Loss не выполнено и изменить Stop Loss позиции не предлагается, иначе проверяется факт изменения цены в сторону увеличения прибыли.

Если ордер Stop Loss у позиции уже установлен, проверяется отступ цены от Stop Loss. Если ордер StopLoss у позиции не установлен, проверяется отступ цены от цены открытия позиции. Если отступ цены больше уровня Stop Loss, то предлагается установить новую цену Stop Loss позиции.

Если условие модификации Stop Loss выполнено и уровень Take Profit не равен нулю, то предлагается установить новую цену Take Profit позиции.

Если инициализация класса эксперта произведена с флагом every_tick=false, то эксперт будет производить торговые операции (и сопровождение торговых позиций) только при появлении нового бара на рабочем символе и таймфрейме. В этом случае установка уровня ордера Take Profit дает возможность закрыть позицию при движении цены в направлении позиции до появления нового бара.

Описание

Класс CTrailingFixedPips реализует алгоритм сопровождения открытых позиций на указанном "расстоянии" от текущей цены (в пунктах).

Декларация

```
class CTrailingFixedPips: public CExpertTrailing
```

Заголовок

```
   #include <Expert\Trailing\CTrailingFixedPips.mqh>
```

Иерархия наследования

CObject
  CExpertBase
    CExpertTrailing
      CTrailingFixedPips

Методы класса по группам

Инициализация	
StopLevel	Устанавливает уровень Stop Loss
ProfitLevel	Устанавливает уровень Take Profit
virtual ValidationSettings	Проверяет корректность настроек

© 2000-2025, MetaQuotes Ltd.

Инициализация	
Методы проверки условий трейлинга	
virtual CheckTrailingStopLong	Определяет необходимость модификации длинной позиции
virtual CheckTrailingStopShort	Определяет необходимость модификации короткой позиции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2025, MetaQuotes Ltd.

StopLevel

Устанавливает уровень сопровождения ордера Stop Loss.

```
void  StopLevel(
   int  stop_level      // уровень
   )
```

Параметры

stop_level

[in] Значение уровня сопровождения ордера Stop Loss в "нормальных" (2/4-знаковых) пунктах.

Примечание

Если задать уровень равным 0, то сопровождение (модификация) ордера не производится.

© 2000-2025, MetaQuotes Ltd.

ProfitLevel

Устанавливает уровень сопровождения ордера Take Profit.

```
void  ProfitLevel(
   int  profit_level      // уровень
   )
```

Параметры

profit_level

[in] Значение уровня сопровождения ордера Take Profit в "нормальных" (2/4-знаковых) пунктах.

Примечание

Если задать уровень равным 0, то сопровождение (модификация) ордера не производится.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Проверяет корректность настроек.

```
virtual bool  ValidationSettings()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Проверяет значения уровней сопровождения. Корректными значениями являются 0 и значения, большие уровня "Минимальный отступ в пунктах от текущей цены для установки Stop ордеров" рабочего символа.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Определяет необходимость модификации длинной позиции.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // указатель
   double&         sl,            // ссылка для Stop Loss
   double&         tp             // ссылка для Take Profit
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Если уровень сопровождения ордера Stop Loss равен нулю, условие не выполнено (уходим). Если ордер Stop Loss позиции уже установлен, его цена принимается за базовую, иначе за базовую цену принимается цена открытия позиции. Если текущая цена Bid выше базовой более чем на уровень сопровождения, то ордер Stop Loss позиции предлагается переместить на расстояние уровня сопровождения ниже текущей цены Bid. Если, при этом, уровень сопровождения ордера Take Profit не равен нулю, то предлагается переместить ордер Take Profit позиции на соответствующее расстояние выше текущей цены Bid.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Определяет необходимость модификации короткой позиции.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // указатель
   double&         sl,           // ссылка для Stop Loss
   double&         tp            // ссылка для Take Profit
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Если уровень сопровождения ордера Stop Loss равен нулю, условие не выполнено (уходим). Если ордер Stop Loss позиции уже установлен, его цена принимается за базовую, иначе за базовую цену принимается цена открытия позиции. Если текущая цена Ask ниже базовой более чем на уровень сопровождения, то ордер Stop Loss позиции предлагается переместить на расстояние уровня сопровождения выше текущей цены Ask. Если, при этом, уровень сопровождения ордера Take Profit не равен нулю, то предлагается переместить ордер Take Profit позиции на соответствующее расстояние ниже текущей цены Ask.

© 2000-2025, MetaQuotes Ltd.

Класс CTrailingMA

Класс CTrailingMA является классом реализации алгоритма сопровождения открытых позиций по значениям скользящей средней.

Описание

Класс CTrailingMA реализует алгоритм сопровождения открытых позиций по значениям скользящей средней с указанными параметрами на предыдущем баре.

Декларация

```
class CTrailingMA: public CExpertTrailing
```

Заголовок

```
   #include <Expert\Trailing\TrailingMA.mqh>
```

Иерархия наследования

CObject
  CExpertBase
    CExpertTrailing
      CTrailingMA

Методы класса по группам

Инициализация	
Period	Устанавливает параметр period скользящей средней
Shift	Устанавливает параметр shift скользящей средней
Method	Устанавливает параметр method скользящей средней
Applied	Устанавливает параметр applied скользящей средней
virtual InitIndicators	Инициализирует индикаторы и таймсерии
virtual ValidationSettings	Проверяет корректность настроек
Методы проверки условий трейлинга	
virtual CheckTrailingStopLong	Определяет необходимость модификации длинной позиции
virtual CheckTrailingStopShort	Определяет необходимость модификации короткой позиции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Period

Устанавливает параметр period для скользящей средней.

```
void  Period(
   int  period      // параметр
   )
```

Параметры

period

[in] Значение параметра period для скользящей средней.

© 2000-2025, MetaQuotes Ltd.

Shift

Устанавливает параметр shift для скользящей средней.

```
void  Shift(
   int  shift      // параметр
   )
```

Параметры

shift

[in] Значение параметра shift для скользящей средней.

© 2000-2025, MetaQuotes Ltd.

Method

Устанавливает параметр method для скользящей средней.

```
void  Method(
   ENUM_MA_METHOD  method      // параметр
   )
```

Параметры

method

[in] Значение параметра method (из перечисления ENUM_MA_METHOD) для скользящей средней.

© 2000-2025, MetaQuotes Ltd.

Applied

Устанавливает параметр applied для скользящей средней.

```
void  Applied(
   ENUM_APPLIED_PRICE  applied      // параметр
   )
```

Параметры

applied

[in] Значение параметра applied (из перечисления ENUM_APPLIED_PRICE) для скользящей средней.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Инициализирует индикаторы и таймсерии.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // указатель
   )
```

Параметры

indicators

[in] Указатель на коллекцию индикаторов и таймсерий, член класса CExpert.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Проверяет корректность настроек.

```
virtual bool  ValidationSettings()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Проверяет значение периода скользящей средней. Корректными являются значения больше 0.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Определяет необходимость модификации длинной позиции.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // указатель
   double&         sl,            // ссылка
   double&         tp             // ссылка
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Вычисляется значение максимально близкого уровня установки Stop Loss. Вычисляется новый уровень установки Stop Loss (исходя из значения средней скользящей на предыдущем баре). Если ордер Stop Loss позиции уже установлен, его цена принимается за базовую, иначе за базовую цену принимается цена открытия позиции. Если новый уровень установки Stop Loss выше базовой цены и ниже максимально близкого уровня установки Stop Loss, то ордер Stop Loss позиции предлагается переместить на новый уровень.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Определяет необходимость модификации короткой позиции.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // указатель
   double&         sl,            // ссылка
   double&         tp             // ссылка
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Вычисляется значение максимально близкого уровня установки Stop Loss. Вычисляется новый уровень установки Stop Loss (исходя из значения средней скользящей на предыдущем баре с учетом спреда). Если ордер Stop Loss позиции уже установлен, его цена принимается за базовую, иначе за базовую цену принимается цена открытия позиции. Если новый уровень установки Stop Loss ниже базовой цены и выше максимально близкого уровня установки Stop Loss, то ордер Stop Loss позиции предлагается переместить на новый уровень.

© 2000-2025, MetaQuotes Ltd.

Класс CTrailingNone

Класс CTrailingNone является классом-заглушкой. Его необходимо использовать для инициализации объекта трейлинга в классе CExpert, в том случае, если торговая стратегия не использует сопровождение открытых позиций.

Описание

Класс CTrailingNone не реализует никаких алгоритмов сопровождения открытых позиций. Методы проверки условий трейлинга всегда возвращают false.

Декларация

```
class CTrailingNone: public CExpertTrailing
```

Заголовок

```
   #include <Expert\Trailing\TrailingNone.mqh>
```

Иерархия наследования

CObject
  CExpertBase
    CExpertTrailing
      CTrailingNone

Методы класса по группам

Методы проверки условий трейлинга	
virtual CheckTrailingStopLong	Метод-заглушка проверки необходимости модификации длинной позиции
virtual CheckTrailingStopShort	Метод-заглушка проверки необходимости модификации короткой позиции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

Методы унаследованные от CExpertTrailing

CheckTrailingStopLong, CheckTrailingStopShort

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Определяет необходимость модификации длинной позиции.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // указатель
   double&         sl,            // ссылка
   double&         tp             // ссылка
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Метод всегда возвращает false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Определяет необходимость модификации короткой позиции.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // указатель
   double&         sl,           // ссылка
   double&         tp            // ссылка
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Метод всегда возвращает false.

© 2000-2025, MetaQuotes Ltd.

Класс CTrailingPSAR

Класс CTrailingPSAR является классом реализации алгоритма сопровождения открытых позиций по значениям индикатора Parabolic SAR.

Описание

Класс CTrailingPSAR реализует алгоритм сопровождения открытых позиций по значениям (на предыдущем баре) индикатора Parabolic SAR с указанными параметрами.

Декларация

```
class CTrailingPSAR: public CExpertTrailing
```

Заголовок

```
   #include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

Иерархия наследования

CObject
  CExpertBase
    CExpertTrailing
      CTrailingPSAR

Методы класса по группам

Инициализация	
Step	Устанавливает параметр step индикатора Parabolic SAR
Maximum	Устанавливает параметр maximum индикатора Parabolic SAR
virtual InitIndicators	Инициализирует индикаторы и таймсерии
Методы проверки условий трейлинга	
virtual CheckTrailingStopLong	Определяет необходимость модификации длинной позиции
virtual CheckTrailingStopShort	Определяет необходимость модификации короткой позиции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Step

Устанавливает параметр step индикатора Parabolic SAR.

```
void  Step(
   double  step      // параметр
   )
```

Параметры

step

[in] Значение параметра step для индикатора Parabolic SAR.

© 2000-2025, MetaQuotes Ltd.

Maximum

Устанавливает параметр maximum индикатора Parabolic SAR.

```
void  Maximum(
   double  maximum      // параметр
   )
```

Параметры

maximum

[in] Значение параметра maximum для индикатора Parabolic SAR.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

Инициализирует индикаторы и таймсерии.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // указатель
   )
```

Параметры

indicators

[in] Указатель на коллекцию индикаторов и таймсерий, член класса CExpert.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

Определяет необходимость модификации длинной позиции.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // указатель
   double&         sl,            // ссылка
   double&         tp             // ссылка
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Вычисляется значение максимально близкого уровня установки Stop Loss. Вычисляется новый уровень установки Stop Loss (исходя из значения индикатора Parabolic SAR на предыдущем баре). Если ордер Stop Loss позиции уже установлен, его цена принимается за базовую, иначе за базовую цену принимается цена открытия позиции. Если новый уровень установки Stop Loss выше базовой цены и ниже максимально близкого уровня установки Stop Loss, то ордер Stop Loss позиции предлагается переместить на новый уровень.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

Определяет необходимость модификации короткой позиции.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // указатель
   double&         sl,            // ссылка
   double&         tp             // ссылка
   )
```

Параметры

position

[in] Указатель на объект CPositionInfo.

sl

[in][out] Ссылка на переменную для размещения цены Stop Loss.

tp

[in][out] Ссылка на переменную для размещения цены Take Profit.

Возвращаемое значение

true - в случае выполнения условия, иначе - false.

Примечание

Вычисляется значение максимально близкого уровня установки Stop Loss. Вычисляется новый уровень установки Stop Loss (исходя из значения индикатора Parabolic SAR на предыдущем баре с учетом спреда). Если ордер Stop Loss позиции уже установлен, его цена принимается за базовую, иначе за базовую цену принимается цена открытия позиции. Если новый уровень установки Stop Loss ниже базовой цены и выше максимально близкого уровня установки Stop Loss, то ордер Stop Loss позиции предлагается переместить на новый уровень.

© 2000-2025, MetaQuotes Ltd.

Набор готовых реализаций алгоритмов управления капиталом и рисками

Этот раздел содержит технические детали работы с классами готовых реализаций алгоритмов управления капиталом и рисками и описание соответствующих компонентов стандартной библиотеки MQL5.

Использование классов готовых реализаций алгоритмов управления капиталом и рисками позволит сэкономить время при разработке (а особенно, проверке) торговых стратегий.

Стандартная библиотека MQL5 (в части набора готовых реализаций алгоритмов управления капиталом и рисками) размещается в рабочем каталоге терминала в папке Include\Expert\Money.

Класс	Описание
CMoneyFixedLot	Класс реализации алгоритма для входа в рынок фиксированным лотом, заданным при настройке
CMoneyFixedMargin	Класс реализации алгоритма для входа в рынок фиксированным уровнем маржи, заданным при настройке
CMoneyFixedRisk	Класс реализации алгоритма для входа в рынок фиксированным уровнем риска, заданным при настройке
CMoneyNone	Класс реализации алгоритма для входа в рынок минимальным лотом
CMoneySizeOptimized	Класс реализации алгоритма для входа в рынок с объемом, определяемом результатами предыдущих сделок

© 2000-2025, MetaQuotes Ltd.

Класс CMoneyFixedLot

Класс CMoneyFixedLot является классом для реализации алгоритма торговли с фиксированным лотом.

Описание

Класс CMoneyFixedLot реализует алгоритм для входа в рынок фиксированным лотом, заданным при настройке.

Декларация

```
class CMoneyFixedLot: public CExpertMoney
```

Заголовок

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

Иерархия наследования

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedLot

Методы класса по группам

Инициализация	
Lots	Устанавливает торговый объем
virtual ValidationSettings	Проверяет корректность настроек
Методы управления капиталом и рисками	
virtual CheckOpenLong	Определяет объем для открытия длинной позиции
virtual CheckOpenShort	Определяет объем для открытия короткой позиции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Методы унаследованные от CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Lots

Устанавливает торговый объем (в лотах).

```
void  Lots(
   double  lots      // количество лотов
   )
```

Параметры

lots

[in] Торговый объем.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Проверяет корректность настроек.

```
virtual bool  ValidationSettings()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Проверяет фиксированный лот, задаваемый при настройке, на предмет попадания в интервал допустимых значений и кратность шагу изменения.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Определяет объем для открытия длинной позиции.

```
virtual double  CheckOpenLong(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объем для открытия длинной позиции.

Примечание

Метод всегда предлагает фиксированный лот, заданный при настройке.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Определяет объем для открытия короткой позиции.

```
virtual double  CheckOpenShort(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объем для открытия короткой позиции.

Примечание

Метод всегда предлагает фиксированный лот, заданный при настройке.

© 2000-2025, MetaQuotes Ltd.

Класс CMoneyFixedMargin

Класс CMoneyFixedMargin является классом для реализации алгоритма торговли с фиксированным уровнем маржи.

Описание

Класс CMoneyFixedMargin реализует алгоритм для входа в рынок фиксированным уровнем маржи, заданным при настройке.

Декларация

```
class CMoneyFixedMargin: public CExpertMoney
```

Заголовок

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

Иерархия наследования

CObject

CExpertBase

CExpertMoney

CMoneyFixedMargin

Методы класса по группам

Методы управления капиталом и рисками	
virtual CheckOpenLong	Определяет объем для открытия длинной позиции
virtual CheckOpenShort	Определяет объем для открытия короткой позиции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Методы унаследованные от CExpertMoney

Percent, ValidationSettings, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Определяет объем для открытия длинной позиции.

```
virtual double  CheckOpenLong(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объём для открытия длинной позиции.

Примечание

Метод определяет объем для открытия длинной позиции с фиксированным уровнем маржи, заданным при настройке. Уровень маржи определяется параметром Percent базового класса CExpertMoney.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Определяет объем для открытия короткой позиции.

```
virtual double  CheckOpenShort(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объем для открытия короткой позиции.

Примечание

Метод определяет объем для открытия короткой позиции с фиксированным уровнем маржи, заданным при настройке. Уровень маржи определяется параметром Percent базового класса CExpertMoney.

© 2000-2025, MetaQuotes Ltd.

Класс CMoneyFixedRisk

Класс CMoneyFixedRisk является классом для реализации алгоритма торговли с фиксированным уровнем риска.

Описание

Класс CMoneyFixedRisk реализует алгоритм для входа в рынок фиксированным уровнем риска, заданным при настройке.

Декларация

```
class CMoneyFixedRisk: public CExpertMoney
```

Заголовок

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

Иерархия наследования

CObject

CExpertBase

CExpertMoney

CMoneyFixedRisk

Методы класса по группам

Методы управления капиталом и рисками	
virtual CheckOpenLong	Определяет объем для открытия длинной позиции
virtual CheckOpenShort	Определяет объем для открытия короткой позиции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Методы унаследованные от CExpertMoney

Percent, ValidationSettings, CheckReverse

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Определяет объем для открытия длинной позиции.

```
virtual double  CheckOpenLong(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объем для открытия длинной позиции.

Примечание

Метод определяет объем для открытия длинной позиции с фиксированным уровнем риска, заданным при настройке. Уровень маржи определяется параметром Percent базового класса CExpertMoney.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Определяет объем для открытия короткой позиции.

```
virtual double  CheckOpenShort(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объем для открытия короткой позиции.

Примечание

Метод определяет объем для открытия короткой позиции с фиксированным уровнем риска, заданным при настройке. Уровень маржи определяется параметром Percent базового класса CExpertMoney.

© 2000-2025, MetaQuotes Ltd.

Класс CMoneyNone

Класс CMoneyNone является классом для реализации "отсутствия" управления капиталом и рисками.

Описание

Класс CMoneyNone реализует алгоритм для входа в рынок минимальным лотом.

Декларация

```
class CMoneyNone: public CExpertMoney
```

Заголовок

```
#include <Expert\Money\MoneyNone.mqh>
```

Иерархия наследования

CObject
  CExpertBase
    CExpertMoney
      CMoneyNone

Методы класса по группам

Инициализация	
virtual ValidationSettings	Проверяет корректность настроек
Методы управления капиталом и рисками	
virtual CheckOpenLong	Определяет объем для открытия длинной позиции
virtual CheckOpenShort	Определяет объем для открытия короткой позиции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Методы унаследованные от CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Проверяет корректность настроек.

```
virtual bool  ValidationSettings()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Метод всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Определяет объём для открытия длинной позиции.

```
virtual double  CheckOpenLong(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объём для открытия длинной позиции.

Примечание

Метод всегда предлагает минимальный лот.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Определяет объем для открытия короткой позиции.

```
virtual double  CheckOpenShort(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объем для открытия короткой позиции.

Примечание

Метод всегда предлагает минимальный лот.

© 2000-2025, MetaQuotes Ltd.

Класс CMoneySizeOptimized

Класс CMoneySizeOptimized является классом для реализации алгоритма управления капиталом и рисками с учетом результатов предыдущих сделок.

Описание

Класс CMoneySizeOptimized реализует алгоритм для входа в рынок с объемом, определяемым результатами предыдущих сделок.

Декларация

```
class CMoneySizeOptimized: public CExpertMoney
```

Заголовок

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

Иерархия наследования

CObject
  CExpertBase
    CExpertMoney
      CMoneySizeOptimized

Методы класса по группам

Инициализация	
DecreaseFactor	Устанавливает значение параметра
virtual ValidationSettings	Проверяет корректность настроек
Методы управления капиталом и рисками	
virtual CheckOpenLong	Определяет объем для открытия длинной позиции
virtual CheckOpenShort	Определяет объем для открытия короткой позиции

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Методы унаследованные от CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

DecreaseFactor

Устанавливает параметр "Фактор снижения".

```
void  DecreaseFactor(
   double  decrease_factor      // фактор снижения
   )
```

Параметры

decrease_factor

[in] Значение параметра "Фактор снижения".

Примечание

Параметр задает коэффициент уменьшения объема открываемой позиции (относительно объема предыдущей позиции), если перед этим была серия убыточных сделок.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

Проверяет корректность настроек.

```
virtual bool  ValidationSettings()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Проверяет на "неотрицательность" значение параметра "Фактор снижения", задаваемое при настройке.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

Определяет объем для открытия длинной позиции.

```
virtual double  CheckOpenLong(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объем для открытия длинной позиции.

Примечание

Метод определяет объем для открытия длинной позиции с учетом результатов предыдущих сделок.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

Определяет объем для открытия короткой позиции.

```
virtual double  CheckOpenShort(
   double  price,      // цена
   double  sl          // цена Stop Loss
   )
```

Параметры

price

[in] Предполагаемая цена открытия.

sl

[in] Предполагаемая цена ордера Stop Loss.

Возвращаемое значение

Объем для открытия короткой позиции.

Примечание

Метод определяет объем для открытия короткой позиции с учетом результатов предыдущих сделок.

© 2000-2025, MetaQuotes Ltd.

Классы для создания панелей индикации и диалогов управления

Этот раздел содержит технические детали работы с классами для создания панелей индикации и диалогов управления и описание соответствующих компонентов Стандартной библиотеки MQL5.

Использование классов для создания панелей индикации и диалогов управления позволит сэкономить время при разработке собственных интерактивных MQL5-приложений, каковыми могут быть эксперты и индикаторы.

Стандартная библиотека MQL5 (в части набора классов для создания панелей индикации и диалогов управления) размещается в рабочем каталоге терминала в папке MQL5\Include\Controls.

Примеры работы с классами можно найти в следующих статьях:

- Как создать графическую панель любой сложности и как это работает
- Улучшаем работу с Панелями: добавляем прозрачность, меняем цвет фона и наследуемся от CAppDialog/CWndClient
- Как быстро добавить панель управления к индикатору и советнику
- Создай свои графические панели на MQL5
- Создание активных панелей управления на MQL5 для торговли

Пример эксперта, иллюстрирующего работу данных классов, находится в папке MQL5\Expert\Examples\Controls.

Вспомогательные структуры	Описание
CRect	Структура прямоугольной области графика
CDateTime	Структура для работы с датой и временем

Базовые классы	Описание
CWnd	Базовый класс для всех элементов управления
CWndObj	Базовый класс для создания панелей индикации и диалогов управления
CWndContainer	Базовый класс для комбинированных элементов управления

Простые элементы управления	Описание
CLabel	Элемент управления на основе объекта графика "Текстовая метка"
CBmpButton	Элемент управления на основе объекта графика "Графическая метка"
CButton	Элемент управления на основе объекта графика "Кнопка"

© 2000-2025, MetaQuotes Ltd.

Простые элементы управления	Описание
CEdit	Элемент управления на основе объекта графика "Поле ввода"
CPanel	Элемент управления на основе объекта графика "Прямоугольная метка"
CPicture	Элемент управления на основе объекта графика "Графическая метка"

Комбинированные элементы управления	Описание
CScroll	Базовый класс полосы прокрутки
CScrollV	Вертикальная полоса прокрутки
CScrollH	Горизонтальная полоса прокрутки
CWndClient	Базовый класс элемента управления, имеющего полосы прокрутки
CListView	Просмотр списка
CComboBox	Поле с выпадающим списком
CCheckBox	Переключатель
CCheckGroup	Групповой переключатель с независимой фиксацией
CRadioButton	Элемент группового переключателя с зависимой фиксацией
CRadioGroup	Групповой переключатель с зависимой фиксацией
CSpinEdit	Поле инкремента/декремента
CDialog	Диалог
CAppDialog	Главный диалог MQL5-программы

© 2000-2025, MetaQuotes Ltd.

Класс CRect

Класс CRect является реализацией класса прямоугольной области графика.

Описание

Класс CRect является программной реализацией прямоугольной области графика в декартовых координатах.

Декларация

```
class CRect
```

Заголовок

```
#include <Controls\Rect.mqh>
```

Методы класса по группам

Свойства	
Left	Получает/устанавливает координату X левой верхней точки области
Top	Получает/устанавливает координату Y левой верхней точки области
Right	Получает/устанавливает координату X правой нижней точки области
Bottom	Получает/устанавливает координату Y правой нижней точки области
Width	Получает/устанавливает ширину области
Height	Получает/устанавливает высоту области
SetBound	Устанавливает новые координаты области
Move	Осуществляет абсолютное перемещение координат области
Shift	Осуществляет относительное перемещение (сдвиг) координат области
Contains	Проверяет факт нахождения точки внутри области
Дополнительные методы	
Format	Получает координаты области в виде строки

© 2000-2025, MetaQuotes Ltd.

Left (метод Get)

Получает координату X левой верхней точки прямоугольной области.

```
int  Left()
```

Возвращаемое значение

Координата X левой верхней точки.

Left (метод Set)

Устанавливает координату X левой верхней точки прямоугольной области.

```
void  Left(
   const int  x      // координата x
   )
```

Параметры

x

[in] Новое значение координаты X левой верхней точки.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Top (метод Get)

Получает координату Y левой верхней точки прямоугольной области.

```
int  Top()
```

Возвращаемое значение

Координата Y левой верхней точки.

Top (метод Set)

Устанавливает координату Y левой верхней точки прямоугольной области.

```
void  Top(
   const int  y      // координата y
   )
```

Параметры

y

[in] Новое значение координаты Y левой верхней точки.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Right (метод Get)

Получает координату X правой нижней точки прямоугольной области.

```
int  Right()
```

Возвращаемое значение

Координата X правой нижней точки.

Right (метод Set)

Устанавливает координату X правой нижней точки прямоугольной области.

```
void  Right(
   const int  x      // координата x
   )
```

Параметры

x

[in] Новое значение координаты X правой нижней точки.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Bottom (метод Get)

Получает координату Y правой нижней точки прямоугольной области.

```
int  Bottom()
```

Возвращаемое значение

Координата Y правой нижней точки.

Bottom (метод Set)

Устанавливает координату Y правой нижней точки прямоугольной области.

```
void  Bottom(
   const int  y      // координата y
   )
```

Параметры

y

[in] Новое значение координаты Y правой нижней точки.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Width (метод Get)

Получает ширину прямоугольной области.

```
int  Width()
```

Возвращаемое значение

Ширина прямоугольной области.

Width (метод Set)

Устанавливает ширину прямоугольной области.

```
virtual bool  Width(
   const int  w      // ширина
   )
```

Параметры

w

[in] Новое значение ширины.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Height (метод Get)

Получает высоту прямоугольной области.

```
int  Height()
```

Возвращаемое значение

Высота прямоугольной области.

Height (метод Set)

Устанавливает высоту прямоугольной области.

```
virtual bool  Height(
   const int  h      // высота
   )
```

Параметры

h

[in] Новое значение высоты.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

SetBound

Устанавливает новые параметры прямоугольной области из координат класса CRect.

```
void  SetBound(
   const & CRect  rect      // класс прямоугольной области
   )
```

Возвращаемое значение

Нет.

SetBound

Устанавливает новые параметры прямоугольной области.

```
void  SetBound(
   const int  l      // left
   const int  t      // topt
   const int  r      // right
   const int  b      // bottom
   )
```

Параметры

l

[in] Координата X левой верхней точки.

t

[in] Координата Y левой верхней точки.

r

[in] Координата X правой нижней точки.

b

[in] Координата Y правой нижней точки.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Move

Осуществляет абсолютное перемещение координат прямоугольной области.

```
void  Move(
   const int  x,      // координата X
   const int  y       // координата Y
   )
```

Параметры

x

[in] Новое значение координаты X.

y

[in] Новое значение координаты Y.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Shift

Осуществляет относительное перемещение (сдвиг) координат прямоугольной области.

```
void  Shift(
   const int  dx,      // смещение по X
   const int  dy       // смещение по Y
   )
```

Параметры

dx

[in] Относительное смещение по координате X.

dy

[in] Относительное смещение по координате Y.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Contains

Проверяет факт нахождения точки с заданными координатами внутри прямоугольной области.

```
bool  Contains(
   const int  x,      // координата X
   const int  y       // координата Y
   )
```

Параметры

x

[in] Координата X точки.

y

[in] Координата Y точки.

Возвращаемое значение

true, если точка находится внутри прямоугольной области (включая границы), иначе - false.

© 2000-2025, MetaQuotes Ltd.

Format

Получает значения координат прямоугольной области в виде строки.

```
string  Format(
   string &  fmt,     // формат
   ) const
```

Параметры

fmt

[in] Строка с описанием формата.

Возвращаемое значение

Строка с координатами прямоугольной области.

© 2000-2025, MetaQuotes Ltd.

Структура CDateTime

Структура CDateTime предназначена для работы с датой и временем.

Описание

Структура CDateTime является наследником MqlDateTime, она используется для работы с датой и временем в элементах управления.

Декларация

```
struct CDateTime
```

Заголовок

```
#include <Tools\DateTime.mqh>
```

Методы

Свойства	
MonthName	Возвращает наименование месяца
ShortMonthName	Возвращает краткое наименование месяца
DayName	Возвращает полное наименование дня недели
ShortDayName	Возвращает краткое наименование дня недели
DaysInMonth	Возвращает количество дней в месяце
Методы установки/получения значений	
DateTime	Возвращает/Устанавливает дату и время
Date	Устанавливает дату
Time	Устанавливает время
Sec	Устанавливает количество секунд
Min	Устанавливает количество минут
Hour	Устанавливает час
Day	Устанавливает день месяца
Mon	Устанавливает месяц
Year	Устанавливает год
Дополнительные методы	
SecDec	Вычитает из даты заданное количество секунд

© 2000-2025, MetaQuotes Ltd.

Свойства	
SecInc	Добавляет к дате заданное количество секунд
MinDec	Вычитает из даты заданное количество минут
MinInc	Добавляет к дате заданное количество минут
HourDec	Вычитает из даты заданное количество часов
HourInc	Добавляет к дате заданное количество часов
DayDec	Вычитает из даты заданное количество дней
DayInc	Добавляет к дате заданное количество дней
MonDec	Вычитает из даты заданное количество месяцев
MonInc	Добавляет к дате заданное количество месяцев
YearDec	Вычитает из даты заданное количество лет
YearInc	Добавляет к дате заданное количество лет

© 2000-2025, MetaQuotes Ltd.

MonthName

Возвращает наименование месяца.

```
string  MonthName()  const
```

Возвращает наименование месяца по номеру.

```
string  MonthName(
   const int       num          // номер месяца
   )  const
```

Параметры

num

[in] Номер месяца (1-12).

Возвращаемое значение

Полное наименование месяца.

© 2000-2025, MetaQuotes Ltd.

ShortMonthName

Возвращает краткое наименование месяца.

```
string  ShortMonthName()  const
```

Возвращает краткое наименование месяца по номеру.

```
string  ShortMonthName(
   const int       num          // номер месяца
   )  const
```

Параметры

num

[in] Номер месяца (1-12).

Возвращаемое значение

Краткое наименование месяца.

© 2000-2025, MetaQuotes Ltd.

DayName

Возвращает полное наименование дня недели.

```
string  DayName()  const
```

Возвращает полное наименование дня недели по номеру.

```
string  DayName(
   const int       num          // номер дня недели
   )  const
```

Параметры

num

[in] Номер дня недели (0-6).

Возвращаемое значение

Полное наименование дня недели.

© 2000-2025, MetaQuotes Ltd.

ShortDayName

Возвращает краткое наименование дня недели.

```
string  ShortDayName()  const
```

Возвращает краткое наименование дня недели по номеру.

```
string  ShortDayName(
   const int       num          // номер дня недели
   )  const
```

Параметры

num

[in] Номер дня недели (0-6).

Возвращаемое значение

Краткое наименование дня недели.

© 2000-2025, MetaQuotes Ltd.

DaysInMonth

Возвращает количество дней в месяце.

```
int  DaysInMonth()  const
```

Возвращаемое значение

Количество дней в месяце.

© 2000-2025, MetaQuotes Ltd.

DateTime (Метод Get)

Возвращает дату и время.

```
datetime  DateTime()
```

Возвращаемое значение

Значение типа datetime.

DateTime (Метод Set datetime)

Устанавливает дату и время с использованием типа datetime.

```
void  DateTime(
   const datetime      value          // дата и время
   )
```

Параметры

value

[in] Значение типа datetime.

Возвращаемое значение

Нет.

DateTime (Метод Set MqlDateTime)

Устанавливает дату и время с использованием типа MqlDateTime.

```
void  DateTime(
   const MqlDateTime   &value         // дата и время
   )
```

Параметры

value

[in] Значение типа MqlDateTime.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Date (Метод Set datetime)

Устанавливает дату с использованием типа datetime.

```
void  Date(
   const datetime      value         // дата
   )
```

Параметры

value

[in] Значение типа datetime.

Возвращаемое значение

Нет.

Date (Метод Set MqlDateTime)

Устанавливает дату с использованием типа MqlDateTime.

```
void  Date(
   const MqlDateTime   &value        // дата
   )
```

Параметры

value

[in] Значение типа MqlDateTime.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Time (Метод Set datetime)

Устанавливает время с использованием типа datetime.

```
void  Time(
   const datetime      value          // время
   )
```

Параметры

value

[in] Значение типа datetime.

Возвращаемое значение

Нет.

Time (Метод Set MqlDateTime)

Устанавливает время с использованием типа MqlDateTime.

```
void  Time(
   const MqlDateTime   &value         // время
   )
```

Параметры

value

[in] Значение типа MqlDateTime.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Sec

Устанавливает количество секунд.

```
void  Sec(
   const int   value        // количество секунд
   )
```

Параметры

value

[in] Количество секунд.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Min

Устанавливает количество минут.

```
void  Min(
   const int    value        // количество минут
   )
```

Параметры

value

[in] Количество минут.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Hour

Устанавливает час.

```
void  Hour(
   const int    value        // час
   )
```

Параметры

value

[in] Час.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Day

Устанавливает день месяца.

```
void  Day(
   const int   value        // день месяца
   )
```

Параметры

value

[in] День месяца.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Mon

Устанавливает месяц.

```
void  Mon(
   const int   value        // номер месяца
   )
```

Параметры

value

[in] Номер месяца.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Year

Устанавливает год.

```
void  Year(
   const int   value        // год
   )
```

Параметры

value

[in] Номер года.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

SecDec

Вычитает из даты заданное количество секунд.

```
void  SecDec(
   int   delta=1        // количество секунд
   )
```

Параметры

delta

[in] Количество секунд.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

SecInc

Добавляет к дате заданное количество секунд.

```
void  SecInc(
   int   delta=1        // количество секунд
   )
```

Параметры

delta

[in] Количество секунд.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

MinDec

Вычитает из даты заданное количество минут.

```
void  MinDec(
   int   delta=1        // количество минут
   )
```

Параметры

delta

[in] Количество минут.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

MinInc

Добавляет к дате заданное количество минут.

```
void  MinInc(
   int   delta=1        // количество минут
   )
```

Параметры

delta

[in] Количество минут.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

HourDec

Вычитает из даты заданное количество часов.

```
void  HourDec(
   int   delta=1        // количество часов
   )
```

Параметры

delta

[in] Количество часов.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

HourInc

Добавляет к дате заданное количество часов.

```
void  HourInc(
   int   delta=1        // количество часов
   )
```

Параметры

delta

[in] Количество часов.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

DayDec

Вычитает из даты заданное количество дней.

```
void  DayDec(
   int   delta=1        // количество дней
   )
```

Параметры

delta

[in] Количество дней.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

DayInc

Добавляет к дате заданное количество дней.

```
void  DayInc(
   int   delta=1        // days
   )
```

Параметры

delta

[in] Количество дней.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

MonDec

Вычитает из даты заданное количество месяцев.

```
void  MonDec(
   int   delta=1        // months
   )
```

Параметры

delta

[in] Количество месяцев.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

MonInc

Добавляет к дате заданное количество месяцев.

```
void  MonInc(
   int   delta=1        // количество месяцев
   )
```

Параметры

delta

[in] Months to add.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

YearDec

Вычитает из даты заданное количество лет.

```
void  YearDec(
   int   delta=1        // количество лет
   )
```

Параметры

delta

[in] Количество лет.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

YearInc

Добавляет к дате заданное количество лет.

```
void  YearInc(
   int   delta=1        // количество лет
   )
```

Параметры

delta

[in] Количество лет.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Класс CWnd

Класс CWnd является базовым классом для всех элементов управления Стандартной библиотеки.

Описание

Класс CWnd является программной реализацией базового класса элемента объекта управления.

Декларация

```
class CWnd : public CObject
```

Заголовок

```
#include <Controls\Wnd.mqh>
```

Иерархия наследования

CObject

CWnd

Прямые потомки

CDragWnd, CWndContainer, CWndObj

Методы класса по группам

Создание и уничтожение	
Create	Создает элемент
Destroy	Уничтожает элемент
Обработчики событий графика	
OnEvent	Обрабатывает все события графика
OnMouseEvent	Обрабатывает только событие графика CHARTEVENT_MOUSE_MOVE
Наименование	
Name	Получает имя элемента
Механизм доступа к контейнеру	
ControlsTotal	Получает количество элементов в контейнере
Control	Получает элемент контейнера по указанному индексу
ControlFind	Получает элемент контейнера по указанному идентификатору
Геометрия	
Rect	Получает координаты элемента

© 2000-2025, MetaQuotes Ltd.

Создание и уничтожение	
Left	Получает/устанавливает X-координату левого верхнего угла элемента
Top	Получает/устанавливает Y-координату левого верхнего угла элемента
Right	Получает/устанавливает X-координату правого нижнего угла элемента
Bottom	Получает/устанавливает Y-координату правого нижнего угла элемента
Width	Получает/устанавливает ширину элемента
Height	Получает/устанавливает высоту элемента
Move	Абсолютное перемещение элемента
Shift	Относительное перемещение элемента
Resize	Изменяет размеры элемента
Contains	Проверяет факт нахождения точки/элемента в области графика, занятой элементом
Выравнивание	
Alignment	Устанавливает параметры выравнивания элемента
Align	Выравнивает элемент в указанной области графика
Идентификация	
Id	Получает/устанавливает идентификатор элемента
Состояние	
IsEnabled	Проверяет способность элемента отвечать на действия пользователя
Enable	Включает способность элемента отвечать на действия пользователя
Disable	Отключает способность элемента отвечать на действия пользователя
IsVisible	Проверяет флаг видимости элемента
Visible	Устанавливает флаг видимости элемента на графике
Show	Отображает элемент управления на графике
Hide	Скрывает элемент управления с графика
IsActive	Проверяет активность элемента
Activate	Делает элемент активным

© 2000-2025, MetaQuotes Ltd.

Создание и уничтожение	
Deactivate	Делает элемент неактивным
Флаги состояния	
StateFlags	Получает/изменяет флаги состояния элемента
StateFlagsSet	Устанавливает флаги состояния элемента
StateFlagsReset	Сбрасывает флаги состояния элемента
Флаги свойств	
PropFlags	Получает/изменяет флаги свойств элемента
PropFlagsSet	Устанавливает флаги свойств элемента
PropFlagsReset	Сбрасывает флаги свойств элемента
Для операций с мышью	
MouseX	Получает/сохраняет координату X курсора мыши
MouseY	Получает/сохраняет координату Y курсора мыши
MouseFlags	Получает/сохраняет состояние кнопок мыши
MouseFocusKill	Отбирает фокус мыши
Обработка внутренних событий	
OnCreate	Обработчик события "Create" элемента управления
OnDestroy	Обработчик события "Destroy" элемента управления
OnMove	Обработчик события "Move" элемента управления
OnResize	Обработчик события "Resize" элемента управления
OnEnable	Обработчик события "Enable" элемента управления
OnDisable	Обработчик события "Disable" элемента управления
OnShow	Обработчик события "Show" элемента управления
OnHide	Обработчик события "Hide" элемента управления
OnActivate	Обработчик события "Activate" элемента управления
OnDeactivate	Обработчик события "Deactivate" элемента управления
OnClick	Обработчик события "Click" элемента управления
OnChange	Обработчик события "Change" элемента управления
Обработка событий мыши	
OnMouseDown	Обработчик события "MouseDown" элемента управления
OnMouseUp	Обработчик события "MouseUp" элемента управления

© 2000-2025, MetaQuotes Ltd.

Создание и уничтожение	
Обработка событий перетаскивания	
OnDragStart	Обработчик события "DragStart" элемента управления
OnDragProcess	Обработчик события "DragProcess" элемента управления
OnDragEnd	Обработчик события "DragEnd" элемента управления
Работа с объектом	
DragObjectCreate	Создает объект перетаскивания
DragObjectDestroy	Уничтожает объект перетаскивания

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата x1
   const int      y1,          // координата y1
   const int      x2,          // координата x2
   const int      y2           // координата y2
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Метод базового класса только сохраняет параметры создания и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

Destroy

Уничтожает элемент управления.

```
virtual bool  Destroy()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр события
   const double&  dparam,      // параметр события
   const string&  sparam       // параметр события
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMouseEvent

Обрабатывает события мыши (событие графика CHARTEVENT_MOUSE_MOVE).

```
virtual bool  OnMouseEvent(
   const int  x,          // координата x
   const int  y,          // координата y
   const int  flags       // флаги
   )
```

Параметры

x

[in] Координата X курсора мышки относительно верхнего левого угла графика.

y

[in] Координата Y курсора мышки относительно верхнего левого угла графика.

flags

[in] Флаги состояния кнопок мыши.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Name

Получает имя объекта управления.

```
string  Name()  const
```

Возвращаемое значение

Имя объекта управления.

© 2000-2025, MetaQuotes Ltd.

ControlsTotal

Получает количество элементов управления в контейнере.

```
int  ControlsTotal()  const
```

Возвращаемое значение

Количество элементов управления в контейнере.

Примечание

Метод базового класса не имеет контейнера элементов, обеспечивая только механизм доступа для своих потомков, и всегда возвращает 0.

© 2000-2025, MetaQuotes Ltd.

Control

Получает элемент управления из контейнера по указанному индексу.

```
CWnd*  Control(
   const int  ind      // индекс
   )  const
```

Параметры

ind

[in] Индекс искомого элемента в контейнере.

Возвращаемое значение

Указатель на элемент управления из контейнера.

Примечание

Метод базового класса не имеет контейнера элементов, обеспечивая только механизм доступа для своих потомков, и всегда возвращает NULL.

© 2000-2025, MetaQuotes Ltd.

ControlFind

Получает элемент управления из контейнера по указанному идентификатору.

```
virtual CWnd*  ControlFind(
   const long  id      // идентификатор
   )
```

Параметры

id

[in] Идентификатор искомого элемента управления.

Возвращаемое значение

Указатель на элемент управления из контейнера.

Примечание

Метод базового класса не имеет контейнера элементов, обеспечивая механизм доступа для своих потомков. При совпадении искомого идентификатора с собственным возвращает указатель на себя (this).

© 2000-2025, MetaQuotes Ltd.

Rect

Получает указатель на объект с координатами элемента управления.

```
const CRect*  Rect()  const
```

Возвращаемое значение

Указатель на объект.

© 2000-2025, MetaQuotes Ltd.

Left (метод Get)

Получает координату X левой верхней точки элемента управления.

```
int  Left()
```

Возвращаемое значение

Координата X левой верхней точки элемента управления.

Left (метод Set)

Устанавливает координату X левой верхней точки элемента управления.

```
virtual void  Left(
   const int  x      // координата
   )
```

Параметры

x

[in] Новое значение координаты X левой верхней точки.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Top (метод Get)

Получает координату Y левой верхней точки элемента управления.

```
int  Top()
```

Возвращаемое значение

Координата Y левой верхней точки элемента управления.

Top (метод Set)

Устанавливает координату Y левой верхней точки элемента управления.

```
virtual void  Top(
   const int  y      // координата y
   )
```

Параметры

y

[in] Новое значение координаты Y левой верхней точки.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Right (метод Get)

Получает координату X правой нижней точки элемента управления.

```
int  Right()
```

Возвращаемое значение

Координата X правой нижней точки элемента управления.

Right (метод Set)

Устанавливает координату X правой нижней точки элемента управления.

```
virtual void  Right(
   const int  x      // координата x
   )
```

Параметры

x

[in] Новое значение координаты X правой нижней точки.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Bottom (метод Get)

Получает координату Y правой нижней точки элемента управления.

```
int  Bottom()
```

Возвращаемое значение

Координата Y правой нижней точки элемента управления.

Bottom (метод Set)

Устанавливает координату Y правой нижней точки элемента управления.

```
virtual void  Bottom(
   const int  y      // координата y
   )
```

Параметры

y

[in] Новое значение координаты Y правой нижней точки.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Width (метод Get)

Получает ширину элемента управления.

```
int  Width()
```

Возвращаемое значение

Ширина элемента управления.

Width (метод Set)

Устанавливает ширину элемента управления.

```
virtual bool  Width(
   const int  w      // ширина
   )
```

Параметры

w

[in] Новое значение ширины.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Height (метод Get)

Получает высоту элемента управления.

```
int  Height()
```

Возвращаемое значение

Высота элемента управления.

Height (метод Set)

Устанавливает высоту элемента управления.

```
virtual bool  Height(
   const int  h      // высота
   )
```

Параметры

h

[in] Новое значение высоты.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Move

Осуществляет абсолютное пермещение элемента управления.

```
virtual bool  Move(
   const int  x,      // координата x
   const int  y       // координата y
   )
```

Параметры

x

[in] Новое значение координаты X левой верхней точки.

y

[in] Новое значение координаты Y левой верхней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Shift

Осуществляет относительное пермещение элемента управления.

```
virtual bool  Shift(
   const int  dx,      // смещение по x
   const int  dy       // смещение по y
   )
```

Параметры

dx

[in] Относительное смещение по координате X.

dy

[in] Относительное смещение по координате Y.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Resize

Изменяет размеры элемента управления.

```
virtual bool  Resize(
   const int  w,
   const int  h
   )
```

Параметры

w

[in] Новая ширина.

h

[in] Новая высота.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Contains

Проверяет факт нахождения точки в области графика, занятой элементом управления.

```
bool  Contains(
   const int  x,      // координата X
   const int  y       // координата Y
   )
```

Параметры

x

[in] Координата X точки.

y

[in] Координата Y точки.

Возвращаемое значение

true - если точка находится внутри области (включая границы), иначе false.

Contains

Проверяет факт нахождения указанного элемента в области графика, занятой элементом управления.

```
bool  Contains(
   const CWnd*  control      // указатель
   )  const
```

Параметры

control

[in] Указатель на элемент.

Возвращаемое значение

true - если элемент находится внутри области (включая границы), иначе false.

© 2000-2025, MetaQuotes Ltd.

Alignment

Устанавливает параметры выравнивания элемента управления.

```
void  Alignment(
   const int  flags,       // флаги
   const int  left,        // отступ
   const int  top,         // отступ
   const int  right,       // отступ
   const int  bottom       // отступ
   )
```

Параметры

flags

[in] Флаги выравнивания элемента управления.

left

[in] Фиксированный отступ от левого края.

top

[in] Фиксированный отступ от верхнего края.

right

[in] Фиксированный отступ от правого края.

bottom

[in] Фиксированный отступ от нижнего края.

Возвращаемое значение

Нет.

Примечание

Флаги выравнивания:

```
enum WND_ALIGN_FLAGS
{
   WND_ALIGN_NONE=0,                                  // нет выравнивания
   WND_ALIGN_LEFT=1,                                  // выравнивание по левой границе
   WND_ALIGN_TOP=2,                                   // выравнивание по верхней грани
   WND_ALIGN_RIGHT=4,                                 // выравнивание по правой границ
   WND_ALIGN_BOTTOM=8,                                // выравнивание по нижней границ
   WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT,  // выравнивание по ширине
   WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,   // выравнивание по высоте
   WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // выравнивание по ширине и высо
   }
```

© 2000-2025, MetaQuotes Ltd.

Align

Выравнивает элемент в указанной области графика.

```
virtual bool  Align(
   const CRect*  rect      // указатель
   )
```

Параметры

rect

[in] Указатель на объект с координатами области графика.

Возвращаемое значение

true-в случае удачи, иначе false.

Примечание

Параметры выравнивания должны быть заданы заранее (по умолчанию выравнивания нет).

© 2000-2025, MetaQuotes Ltd.

Id (метод Get)

Получает идентификатор элемента управления.

```
long  Id()  const
```

Возвращаемое значение

Значение идентификатора.

Id (метод Set)

Устанавливает значение идентификатора элемента управления.

```
virtual long  Id(
   const long  id      // идентификатор
   )
```

Параметры

x

[in] Новое значение идентификатора элемента управления.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

IsEnabled

Проверяет способность элемента управления отвечать на действия пользователя.

```
bool  IsEnabled()  const
```

Возвращаемое значение

true - в случае, если элемент управления способен отвечать на действия пользователя, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Enable

Включает способность элемента отвечать на действия пользователя.

```
virtual bool  Enable()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

При включенном режиме элемент отрабатывает внешние события.

© 2000-2025, MetaQuotes Ltd.

Disable

Отключает способность элемента отвечать на действия пользователя.

```
virtual bool  Disable()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

При отключенном режиме элемент не отрабатывает внешние события.

© 2000-2025, MetaQuotes Ltd.

IsVisible

Проверяет видимость элемента управления.

```
bool  IsVisible()  const
```

Возвращаемое значение

true - в случае, если элемент управления отображается на графике, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Visible

Устанавливает флаг видимости элемента управления в указанное состояние.

```
virtual bool  Visible(
   const bool  flag      // флаг
   )
```

Параметры

flag

[in] Новое состояние флага.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Show

Делает элемент управления видимым.

```
virtual bool  Show()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Hide

Делает элемент управления невидимым.

```
virtual bool  Hide()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

IsActive

Проверяет активность элемента управления.

```
bool  IsActive()  const
```

Возвращаемое значение

true - в случае, если элемент управления активен, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Activate

Делает элемент управления активным.

```
virtual bool  Activate()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Элемент становится активным при наведении на него курсора мыши.

© 2000-2025, MetaQuotes Ltd.

Deactivate

Делает элемент управления неактивным.

```
virtual bool  Deactivate()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Элемент становится неактивным при выведении с него курсора мыши.

© 2000-2025, MetaQuotes Ltd.

StateFlags (метод Get)

Получает флаги свойств элемента управления.

```
int  StateFlags()
```

Возвращаемое значение

Флаги состояния элемента управления.

StateFlags (метод Set)

Устанавливает флаги состояния элемента управления.

```
virtual void  StateFlags(
   const int  flags      // флаги
   )
```

Параметры

flags

[in] Новая комбинация флагов состояния.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

StateFlagsSet

Устанавливает флаги свойств элемента управления.

```
virtual void  StateFlagsSet(
   const int  flags      // флаги
   )
```

Параметры

flags

[in] Комбинация флагов свойств, подлежащих установке.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

StateFlagsReset

Сбрасывает флаги свойств элемента управления.

```
virtual void  StateFlagsReset(
   const int  flags      // флаги
   )
```

Параметры

flags

[in] Комбинация флагов свойств, подлежащих сбросу.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

PropFlags (метод Get)

Получает флаги свойств элемента управления.

```
void  PropFlags(
   const int  flags      // флаги
   )
```

Возвращаемое значение

Флаги свойств элемента управления.

PropFlags (метод Set)

Устанавливает флаги свойств элемента управления.

```
virtual void  PropFlags(
   const int  flags      // флаги
   )
```

Параметры

flags

[in] Новая комбинация флагов свойств.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

PropFlagsSet

Устанавливает флаги свойств элемента управления.

```
virtual void  PropFlagsSet(
   const int  flags      // флаги
   )
```

Параметры

flags

[in] Комбинация флагов свойств, подлежащих установке.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

PropFlagsReset

Сбрасывает флаги свойств элемента управления.

```
virtual void  PropFlagsReset(
   const int  flags      // флаги
   )
```

Параметры

flags

[in] Комбинация флагов свойств, подлежащих сбросу.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

MouseX (метод Set)

Сохраняет координату X курсора мыши.

```
void  MouseX(
   const int  value      // координата
   )
```

Параметры

value

[in] Координата X курсора мыши.

Возвращаемое значение

Нет.

MouseX (метод Get)

Получает сохраненную координату X курсора мыши.

```
int  MouseX()
```

Возвращаемое значение

Координата X курсора мыши.

© 2000-2025, MetaQuotes Ltd.

MouseY (метод Set)

Сохраняет координату Y курсора мыши.

```
void  MouseY(
   const int  value      // координата
   )
```

Параметры

value

[in] Координата Y курсора мыши.

Возвращаемое значение

Нет.

MouseY (метод Get)

Получает сохраненную координату Y курсора мыши.

```
int  MouseY()
```

Возвращаемое значение

Координата Y курсора мыши.

© 2000-2025, MetaQuotes Ltd.

MouseFlags (метод Set)

Сохраняет состояние кнопок мыши.

```
virtual void  MouseFlags(
   const int  value      // состояние
   )
```

Параметры

value

[in] Состояние кнопок мыши.

Возвращаемое значение

Нет.

MouseFlags (метод Get)

Получает сохраненное состояние кнопок мыши.

```
int  MouseFlags()
```

Возвращаемое значение

Состояние кнопок мыши.

© 2000-2025, MetaQuotes Ltd.

MouseFocusKill

Очищает сохраненное состояние мыши и деактивирует элемент управления.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // идентификатор
   )
```

Параметры

id=CONTROLS_INVALID_ID

[in] Идентификатор элемента управления, получившего фокус мыши.

Возвращаемое значение

Результат выполнения деактивации элемента управления.

© 2000-2025, MetaQuotes Ltd.

OnCreate

Виртуальный обработчик внутреннего события "Create" (создание) элемента управления.

```
virtual bool  OnCreate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnDestroy

Виртуальный обработчик внутреннего события "Destroy" (уничтожение) элемента управления.

```
virtual bool  OnDestroy()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnMove

Виртуальный обработчик внутреннего события "Move" (перемещение) элемента управления.

```
virtual bool  OnMove()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnResize

Виртуальный обработчик внутреннего события "Resize" (изменение размеров) элемента управления.

```
virtual bool  OnResize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnEnable

Виртуальный обработчик внутреннего события "Enable" (включение способности отвечать на действия пользователя) элемента управления.

```
virtual bool  OnEnable()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnDisable

Виртуальный обработчик внутреннего события "Disable" (отключение способности отвечать на действия пользователя) элемента управления.

```
virtual bool  OnDisable()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnShow

Виртуальный обработчик внутреннего события "Show" (отображение) элемента управления.

```
virtual bool  OnShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) элемента управления.

```
virtual bool  OnHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnActivate

Виртуальный обработчик внутреннего события "Activate" (активизация) элемента управления.

```
virtual bool  OnActivate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

Виртуальный обработчик внутреннего события "Deactivate" (деактивизация) элемента управления.

```
virtual bool  OnDeactivate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnClick

Виртуальный обработчик внутреннего события "Click" (клик мыши) элемента управления.

```
virtual bool  OnClick()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChange

Виртуальный обработчик внутреннего события "Change" (изменение) элемента управления.

```
virtual bool  OnChange()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

Виртуальный обработчик события мыши "MouseDown" (нажатие кнопки мыши) элемента управления.

```
virtual bool  OnMouseDown()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "MouseDown" происходит при нажатии левой кнопки мыши, если указатель мыши находится на элементе управления.

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

Виртуальный обработчик события мыши "MouseUp" (отпускание кнопки мыши) элемента управления.

```
virtual bool  OnMouseUp()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "MouseUp" происходит при отпускании левой кнопки мыши, если указатель мыши находится на элементе управления.

© 2000-2025, MetaQuotes Ltd.

OnDragStart

Виртуальный обработчик события "DragStart" (начало операции перетаскивания) элемента управления.

```
virtual bool  OnDragStart()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "DragStart" происходит при начале операции перетаскивания элемента управления.

© 2000-2025, MetaQuotes Ltd.

OnDragProcess

Виртуальный обработчик события "DragProcess" (операция перетаскивания) элемента управления.

```
virtual bool  OnDragProcess(
   const int  x,      // координата x
   const int  y       // координата y
   )
```

Параметры

x

[in] Текущее значение координаты X курсора мыши.

y

[in] Текущее значение координаты Y курсора мыши.

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "DragProcess" происходит при перемещении элемента управления.

© 2000-2025, MetaQuotes Ltd.

OnDragEnd

Виртуальный обработчик события "DragEnd" (завершение операции перетаскивания) элемента управления.

```
virtual bool  OnDragEnd()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "DragEnd" происходит по завершении операции перетаскивания элемента управления.

© 2000-2025, MetaQuotes Ltd.

DragObjectCreate

Создает объект перетаскивания.

```
virtual bool  DragObjectCreate()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

DragObjectDestroy

Уничтожает объект перетаскивания.

```
virtual bool  DragObjectDestroy()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CWndObj

Класс CWndObj является базовым классом для простых (основанных на объектах графика) элементов управления Стандартной библиотеки.

Описание

Класс CWndObj является программной реализацией основы простого элемента управления.

Декларация

```
class CWndObj : public CWnd
```

Заголовок

```
#include <Controls\WndObj.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndObj

Прямые потомки

CBmpButton, CButton, CEdit, CLabel, CPanel, CPicture

Методы класса по группам

Обработчики событий графика	
OnEvent	Обрабатывает все события графика
Параметры	
Text	Получает/устанавливает значение свойства OBJPROP_TEXT объекта графика
Color	Получает/устанавливает значение свойства OBJPROP_COLOR объекта графика
ColorBackground	Получает/устанавливает значение свойства OBJPROP_BGCOLOR объекта графика
ColorBorder	Получает/устанавливает значение свойства OBJPROP_BORDER_COLOR объекта графика
Font	Получает/устанавливает значение свойства OBJPROP_FONT объекта графика
FontSize	Получает/устанавливает значение свойства OBJPROP_FONTSIZE объекта графика

© 2000-2025, MetaQuotes Ltd.

Обработчики событий графика	
ZOrder	Получает/устанавливает значение свойства OBJPROP_ZORDER объекта графика
Обработка событий объектов графика	
OnObjectCreate	Обработчик события CHARTEVENT_OBJECT_CREATE объекта графика
OnObjectChange	Обработчик события CHARTEVENT_OBJECT_CHANGE объекта графика
OnObjectDelete	Обработчик события CHARTEVENT_OBJECT_DELETE объекта графика
OnObjectDrag	Обработчик события CHARTEVENT_OBJECT_DRAG объекта графика
Обработка изменения параметров	
OnSetText	Обработчик события "SetText" элемента управления
OnSetColor	Обработчик события "SetColor" элемента управления
OnSetColorBackground	Обработчик события "SetColorBackground" элемента управления
OnSetFont	Обработчик события "SetFont" элемента управления
OnSetFontSize	Обработчик события "SetFontSize" элемента управления
OnSetZOrder	Обработчик события "SetZOrder" элемента управления
Обработка внутренних событий	
OnDestroy	Обработчик внутреннего события "Destroy" элемента управления
OnChange	Обработчик внутреннего события "Change" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Text (метод Get)

Получает значение свойства OBJPROP_TEXT (текст) объекта графика.

```
string  Text()
```

Возвращаемое значение

Значение свойства OBJPROP_TEXT объекта графика.

Text (метод Set)

Устанавливает значение свойства OBJPROP_TEXT (текст) объекта графика.

```
bool  Text(
   const string  value      // значение
   )
```

Параметры

value

[in] Новое значение свойства OBJPROP_TEXT.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Color (метод Get)

Получает значение свойства OBJPROP_COLOR (цвет) объекта графика.

```
color  Color()
```

Возвращаемое значение

Значение свойства OBJPROP_COLOR объекта графика.

Color (метод Set)

Устанавливает значение свойства OBJPROP_COLOR (цвет) объекта графика.

```
bool  Color(
   const color  value      // значение
   )
```

Параметры

value

[in] Новое значение свойства OBJPROP_COLOR.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ColorBackground (метод Get)

Получает значение свойства OBJPROP_BGCOLOR (цвет фона) объекта графика.

```
color  ColorBackground()
```

Возвращаемое значение

Значение свойства OBJPROP_BGCOLOR объекта графика.

ColorBackground (метод Set)

Устанавливает значение свойства OBJPROP_BGCOLOR (цвет фона) объекта графика.

```
bool  ColorBackground(
   const color  value      // значение
   )
```

Параметры

value

[in] Новое значение свойства OBJPROP_BGCOLOR.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ColorBorder (метод Get)

Получает значение свойства OBJPROP_BORDER_COLOR (цвет рамки) объекта графика.

```
color  ColorBorder()
```

Возвращаемое значение

Значение свойства OBJPROP_BORDER_COLOR объекта графика.

ColorBorder (метод Set)

Устанавливает значение свойства OBJPROP_BORDER_COLOR (цвет рамки) объекта графика.

```
bool  ColorBorder(
   const color  value      // значение
   )
```

Параметры

value

[in] Новое значение свойства OBJPROP_BORDER_COLOR.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Font (метод Get)

Получает значение свойства OBJPROP_FONT (шрифт) объекта графика.

```
string  Font()
```

Возвращаемое значение

Значение свойства OBJPROP_FONT объекта графика.

Font (метод Set)

Устанавливает значение свойства OBJPROP_FONT (шрифт) объекта графика.

```
bool  Font(
   const string  value      // значение
   )
```

Параметры

value

[in] Новое значение свойства OBJPROP_FONT.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

FontSize (метод Get)

Получает значение свойства OBJPROP_FONTSIZE (размер шрифта) объекта графика.

```
int  FontSize()
```

Возвращаемое значение

Значение свойства OBJPROP_FONTSIZE объекта графика.

FontSize (метод Set)

Устанавливает значение свойства OBJPROP_FONTSIZE (размер шрифта) объекта графика.

```
bool  FontSize(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение свойства OBJPROP_FONTSIZE.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ZOrder (метод Get)

Получает значение свойства OBJPROP_ZORDER объекта графика.

```
long  ZOrder()
```

Возвращаемое значение

Значение свойства OBJPROP_ZORDER объекта графика.

ZOrder (метод Set)

Устанавливает значение свойства OBJPROP_ZORDER объекта графика.

```
bool  ZOrder(
   const long  value      // значение
   )
```

Параметры

value

[in] Новое значение свойства OBJPROP_ZORDER.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnObjectCreate

Виртуальный обработчик события CHARTEVENT_OBJECT_CREATE объекта графика.

```
virtual bool  OnObjectCreate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnObjectChange

Виртуальный обработчик события CHARTEVENT_OBJECT_CHANGE объекта графика.

```
virtual bool  OnObjectChange()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnObjectDelete

Виртуальный обработчик события CHARTEVENT_OBJECT_DELETE объекта графика.

```
virtual bool  OnObjectDelete()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnObjectDrag

Виртуальный обработчик события CHARTEVENT_OBJECT_DRAG объекта графика.

```
virtual bool  OnObjectDrag()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetText

Виртуальный обработчик события "SetText" (изменение свойства OBJPROP_TEXT) элемента управления.

```
virtual bool  OnSetText()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

Виртуальный обработчик события "SetColor" (изменение свойства OBJPROP_COLOR) элемента управления.

```
virtual bool  OnSetColor()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

Виртуальный обработчик события "SetColorBackground" (изменение свойства OBJPROP_BGCOLOR) элемента управления.

```
virtual bool  OnSetColorBackground()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

Виртуальный обработчик события "SetFont" (изменение свойства OBJPROP_FONT) элемента управления.

```
virtual bool  OnSetFont()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

Виртуальный обработчик события "SetFontSize" (изменение свойства OBJPROP_FONTSIZE) элемента управления.

```
virtual bool  OnSetFontSize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

Виртуальный обработчик события "SetZOrder" (изменение параметра OBJPROP_ZORDER) элемента управления.

```
virtual bool  OnSetZOrder()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnDestroy

Виртуальный обработчик внутреннего события "Destroy" (уничтожение) элемента управления.

```
virtual bool  OnDestroy()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChange

Виртуальный обработчик внутреннего события "Change" (изменение) элемента управления.

```
virtual bool  OnChange()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CWndContainer

Класс CWndContainer является базовым классом для группы элементов управления Стандартной библиотеки.

Описание

Класс CWndContainer является программной реализацией группы функционально связанных элементов управления.

Декларация

```
class CWndContainer : public CWnd
```

Заголовок

```
#include <Controls\WndContainer.mqh>
```

Иерархия наследования

CObject

CWnd

CWndContainer

Прямые потомки

CCheckBox, CComboBox, CDateDropList, CDatePicker, CDialog, CRadioButton, CScroll, CSpinEdit, CWndClient

Методы класса по группам

Уничтожение	
Destroy	Уничтожает группу элементов
Обработчики событий графика	
OnEvent	Обрабатывает все события графика
OnMouseEvent	Обрабатывает только событие графика CHARTEVENT_MOUSE_MOVE
Доступ к содержимому контейнера	
ControlsTotal	Получает количество элементов в контейнере
Control	Элемент контейнера по указанному индексу
ControlFind	Получает элемент контейнера по указанному идентификатору
Наполнение	
Add	Добавляет элемент в группу (контейнер)

© 2000-2025, MetaQuotes Ltd.

Уничтожение	
Delete	Удаляет элемент из группы (контейнера)
Геометрия	
Move	Осуществляет абсолютное перемещение группы элементов
Shift	Осуществляет относительное перемещение группы элементов
Идентификация	
Id	Устанавливает идентификаторы элементов группы
Состояние	
Enable	Включает способность группы элементов отвечать на действия пользователя
Disable	Отключает способность группы элементов отвечать на действия пользователя
Show	Делает группу элементов видимой
Hide	Делает группу элементов невидимой
Для операций с мышью	
MouseFocusKill	Отбирает фокус мыши
Работа с файлами	
Save	Запись информации группы в файл
Load	Чтение информации группы из файла
Обработка внутренних событий	
OnResize	Обработчик события "Resize" группы элементов
OnActivate	Обработчик события "Activate" группы элементов
OnDeactivate	Обработчик события "Deactivate" группы элементов

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Destroy

Уничтожает группу элементов управления.

```
virtual bool  Destroy()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMouseEvent

Обрабатывает события мыши.

```
virtual bool  OnMouseEvent(
   const int  x,          // координата
   const int  y,          // координата
   const int  flags       // флаги
   )
```

Параметры

x

[in] Координата X курсора мышки относительно верхнего левого угла графика.

y

[in] Координата Y курсора мышки относительно верхнего левого угла графика.

flags

[in] Флаги состояния кнопок мыши.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ControlsTotal

Получает количество элементов управления в контейнере.

```
int  ControlsTotal()  const
```

Возвращаемое значение

Количество элементов управления в контейнере.

© 2000-2025, MetaQuotes Ltd.

Control

Получает элемент управления контейнера по указанному индексу.

```
CWnd*  Control(
   const int  ind      // индекс
   )  const
```

Параметры

ind

[in] Индекс искомого элемента в контейнере.

Возвращаемое значение

Указатель на элемент управления, либо NULL, если элемент не найден.

© 2000-2025, MetaQuotes Ltd.

ControlFind

Получает элемент управления контейнера по указанному идентификатору.

```
virtual CWnd*  ControlFind(
   const long  id      // идентификатор
   )
```

Параметры

id

[in] Идентификатор искомого элемента управления.

Возвращаемое значение

Указатель на элемент управления, либо NULL, если элемент не найден.

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент в группу (контейнер).

```
bool  Add(
   CWnd&  control      // ссылка
   )
```

Параметры

control

[in] Ссылка на добавляемый элемент управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Delete

Удаляет элемент из группы (контейнера).

```
bool  Delete(
   CWnd&  control      // ссылка
   )
```

Параметры

control

[in] Ссылка на удаляемый элемент управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Move

Осуществляет абсолютное пермещение группы элементов.

```
virtual bool  Move(
   const int  x,      // координата X
   const int  y       // координата Y
   )
```

Параметры

x

[in] Новое значение координаты X левой верхней точки.

y

[in] Новое значение координаты Y левой верхней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Shift

Осуществляет относительное пермещение группы элементов.

```
virtual bool  Shift(
   const int  dx,      // смещение по X
   const int  dy       // смещение по Y
   )
```

Параметры

dx

[in] Относительное смещение по координате X.

dy

[in] Относительное смещение по координате Y.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Id

Раздает идентификаторы элементам группы.

```
virtual long  Id(
   const long  id      // идентификатор
   )
```

Параметры

id

[in] Базовый идентификатор группы.

Возвращаемое значение

Количество идентификаторов, использованных группой элементов.

© 2000-2025, MetaQuotes Ltd.

Enable

Включает способность группы элементов отвечать на действия пользователя.

```
virtual bool  Enable()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Disable

Отключает способность группы элементов отвечать на действия пользователя.

```
virtual bool  Disable()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Show

Делает группу элементов видимой.

```
virtual bool  Show()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Hide

Делает группу элементов невидимой.

```
virtual bool  Hide()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

MouseFocusKill

Очищает сохраненное состояние мыши и деактивирует группу элементов.

```
virtual bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // идентификатор
   )
```

Параметры

id=CONTROLS_INVALID_ID

[in] Идентификатор элемента управления, получившего фокус мыши.

Возвращаемое значение

Результат выполнения деактивации группы элементов.

© 2000-2025, MetaQuotes Ltd.

Save

Записывает информацию группы в файл.

```
virtual bool  Save(
   const int  file_handle      // хендл
   )
```

Параметры

file_handle

[in] Хендл бинарного файла, ранее открытого для записи.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Load

Читает информацию группы из файл.

```
virtual bool  Load(
   const int  file_handle      // хендл
   )
```

Параметры

file_handle

[in] Хендл бинарного файла, ранее открытого для чтения.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnResize

Виртуальный обработчик внутреннего события "Resize" (изменение размеров) элемента управления.

```
virtual bool  OnResize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnActivate

Виртуальный обработчик внутреннего события "Activate" (активизация) элемента управления.

```
virtual bool  OnActivate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

Виртуальный обработчик внутреннего события "Deactivate" (деакивизация) элемента управления.

```
virtual bool  OnDeactivate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

Класс CLabel

Класс CLabel является классом простого элемента управления на основе объекта "Текстовая метка".

Описание

Класс CLabel предназначен для создания простейших нередактируемых текстовых полей.

Декларация

```
class CLabel : public CWndObj
```

Заголовок

```
#include <Controls\Label.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndObj
      CLabel

Результат работы представленного ниже кода:



Методы класса по группам

Создание	
Create	Создает элемент управления

© 2000-2025, MetaQuotes Ltd.

Создание	
Обработка изменения параметров	
OnSetText	Обработчик события "SetText" элемента управления
OnSetColor	Обработчик события "SetColor" элемента управления
OnSetFont	Обработчик события "SetFont" элемента управления
OnSetFontSize	Обработчик события "SetFontSize" элемента управления
Обработка внутренних событий	
OnCreate	Обработчик внутреннего события "Create" элемента управления
OnShow	Обработчик внутреннего события "Show" элемента управления
OnHide	Обработчик внутреннего события "Hide" элемента управления
OnMove	Обработчик внутреннего события "Move" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Пример создания панели с текстовой меткой:

```
//+------------------------------------------------------------------+
//|                                                ControlsLabel.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 "Control Panels and Dialogs. Demonstration class CLabel"
#include <Controls\Dialog.mqh>
#include <Controls\Label.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 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 allow
#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                                                      |
```

© 2000-2025, 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                                   |
```

© 2000-2025, 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CLabel.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата x1
   const int      y1,          // координата y1
   const int      x2,          // координата x2
   const int      y2           // координата y2
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetText

Виртуальный обработчик события "SetText" (изменение свойства OBJPROP_TEXT) элемента управления CLabel.

```
virtual bool  OnSetText()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

Виртуальный обработчик события "SetColor" (изменение свойства OBJPROP_COLOR) элемента управления CLabel.

```
virtual bool  OnSetColor()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

Виртуальный обработчик события "SetFont" (изменение свойства OBJPROP_FONT) элемента управления CLabel.

```
virtual bool  OnSetFont()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

Виртуальный обработчик события "SetFontSize" (изменение свойства OBJPROP_FONTSIZE) элемента управления CLabel.

```
virtual bool  OnSetFontSize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

Виртуальный обработчик внутреннего события "Create" (создание) элемента управления CLabel.

```
virtual bool  OnCreate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnShow

Виртуальный обработчик внутреннего события "Show" (отображение) элемента управления CLabel.

```
virtual bool  OnShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) элемента управления CLabel.

```
virtual bool  OnHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMove

Виртуальный обработчик внутреннего события "Move" (перемещение) элемента управления CLabel.

```
virtual bool  OnMove()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CBmpButton

Класс CBmpButton является классом простого элемента управления на основе объекта "Графическая метка".

Описание

Класс CBmpButton предназначен для создания кнопки с графическим изображением.

Декларация

```
class CBmpButton : public CWndObj
```

Заголовок

```
#include <Controls\BmpButton.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndObj
      CBmpButton

Результат работы представленного ниже кода:



Методы класса по группам

Создание	
Create	Создает элемент управления

© 2000-2025, MetaQuotes Ltd.

Создание	
Параметры элемента управления	
Border	Получает/устанавливает свойство "Border" элемента управления
BmpNames	Устанавливает имена bmp-файлов для отображения элемента управления
BmpOffName	Получает/устанавливает имя bmp-файла отображения элемента в состоянии OFF
BmpOnName	Получает/устанавливает имя bmp-файла отображения элемента в состоянии ON
BmpPassiveName	Получает/устанавливает имя bmp-файла для отображения элемента в неактивном состоянии
BmpActiveName	Получает/устанавливает имя bmp-файла для отображения элемента в активном состоянии
Свойства	
Pressed	Получает/устанавливает состояние элемента управления
Locking	Получает/устанавливает значение свойства "Locking" элемента управления
Обработка внутренних событий	
OnSetZOrder	Обработчик внутреннего события "SetZOrder" элемента управления
OnCreate	Обработчик внутреннего события "Create" элемента управления
OnShow	Обработчик внутреннего события "Show" элемента управления
OnHide	Обработчик внутреннего события "Hide" элемента управления
OnMove	Обработчик внутреннего события "Move" элемента управления
OnChange	Обработчик внутреннего события "Change" элемента управления
OnActivate	Обработчик внутреннего события "Activate" элемента управления
OnDeactivate	Обработчик внутреннего события "Deactivate" элемента управления
OnMouseDown	Обработчик внутреннего события "MouseDown" элемента управления

© 2000-2025, MetaQuotes Ltd.

Создание	
OnMouseUp	Обработчик внутреннего события "MouseUp" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Пример создания панели с графической меткой

```
//+------------------------------------------------------------------+
//|                                            ControlsBmpButton.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 "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 allow
#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
```

© 2000-2025, MetaQuotes Ltd.

```
#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)
  {
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 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);
  }
//+------------------------------------------------------------------+
//| 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
```

© 2000-2025, MetaQuotes Ltd.

```
   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__);
  }
//+------------------------------------------------------------------+
//| 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
```

© 2000-2025, MetaQuotes Ltd.

```
   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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CBmpButton.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата x1
   const int      y1,          // координата y1
   const int      x2,          // координата x2
   const int      y2           // координата y2
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Border (метод Get)

Получает значение параметра "Border" (ширина рамки) элемента управления CBmpButton.

```
int  Border()  const
```

Возвращаемое значение

Значение параметра "Border".

Border (метод Set)

Устанавливает значение параметра "Border" (ширина рамки) элемента управления CBmpButton.

```
bool  Border(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение параметра "Border".

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

BmpNames

Устанавливает имена bmp-файлов для отображения элемента управления CBmpButton.

```
bool  BmpNames(
   const string  off="",      // имя файла
   const string  on=""        // имя файла
   )
```

Параметры

off=""

[in] Имя bmp-файла для отображения элемента в состоянии OFF.

on=""

[in] Имя bmp-файла для отображения элемента в состоянии ON.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

BmpOffName (метод Get)

Получает имя bmp-файла отображения элемента в состоянии OFF.

```
string  BmpOffName()  const
```

Возвращаемое значение

Имя bmp-файла отображения элемента в состоянии OFF.

BmpOffName (метод Set)

Устанавливает имя bmp-файла для отображения элемента в состоянии OFF.

```
bool  BmpOffName(
   const string  name      // имя файла
   )
```

Параметры

name

[in] Имя bmp-файла для отображения элемента в состоянии OFF.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

BmpOnName (метод Get)

Получает имя bmp-файла отображения элемента в состоянии ON.

```
string  BmpOnName()  const
```

Возвращаемое значение

Имя bmp-файла отображения элемента в состоянии ON.

BmpOnName (метод Set)

Устанавливает имя bmp-файла для отображения элемента в состоянии ON.

```
bool  BmpOnName(
   const string  name       // имя файла
   )
```

Параметры

name

[in] Имя bmp-файла для отображения элемента в состоянии ON.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

BmpPassiveName (метод Get)

Получает имя bmp-файла для отображения элемента управления CBmpButton в неактивном состоянии.

```
string  BmpPassiveName()  const
```

Возвращаемое значение

Имя bmp-файла для отображения элемента в неактивном состоянии.

BmpPassiveName (метод Set)

Устанавливает имя bmp-файла для отображения элемента управления CBmpButton в неактивном состоянии.

```
bool  BmpPassiveName(
   const string  name      // имя файла
   )
```

Параметры

name

[in] Имя bmp-файла для отображения элемента в неактивном состоянии.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

BmpActiveName (метод Get)

Получает имя bmp-файла для отображения элемента управления CBmpButton в активном состоянии.

```
string  BmpActiveName()  const
```

Возвращаемое значение

Имя bmp-файла для отображения элемента в активном состоянии.

Примечание

Элемент управления является активным, если указатель мыши находится на элементе управления.

BmpActiveName (метод Set)

Устанавливает имя bmp-файла для отображения элемента CBmpButton в активном состоянии.

```
bool  BmpActiveName(
   const string  name      // имя файла
   )
```

Параметры

name

[in] Имя bmp-файла для отображения элемента в активном состоянии.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Pressed (метод Get)

Получает состояние (свойство Pressed) элемента управления CBmpButton.

```
bool  Pressed()  const
```

Возвращаемое значение

Состояние элемента.

Pressed (метод Set)

Устанавливает состояние (свойство Pressed) элемента управления CBmpButton.

```
bool  Pressed(
   const bool  pressed      // состояние
   )
```

Параметры

pressed

[in] Новое состояние элемента.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Locking (метод Get)

Получает значение свойства "Locking" (фиксация) элемента управления CBmpButton.

```
bool  Locking()  const
```

Возвращаемое значение

Значение свойства "Locking".

Locking (метод Set)

Устанавливает свойство "Locking" (фиксация) элемента управления CBmpButton.

```
void  Locking(
   const bool  locking      // значение
   )
```

Параметры

locking

[in] Новое значение свойства "Locking".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

Виртуальный обработчик события "SetZOrder" (изменение параметра OBJPROP_ZORDER) элемента управления CBmpButton.

```
virtual bool  OnSetZOrder()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

Виртуальный обработчик внутреннего события "Create" (создание) элемента управления CBmpButton.

```
virtual bool  OnCreate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnShow

Виртуальный обработчик внутреннего события "Show" (отображение) элемента управления CBmpButton.

```
virtual bool  OnShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) элемента управления CBmpButon.

```
virtual bool  OnHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMove

Виртуальный обработчик внутреннего события "Move" (перемещение) элемента управления CBmpButton.

```
virtual bool  OnMove()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChange

Виртуальный обработчик внутреннего события "Change" (изменение) элемента управления CBmpButton.

```
virtual bool  OnChange()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnActivate

Виртуальный обработчик внутреннего события "Activate" (активизация) элемента управления CBmpButton.

```
virtual bool  OnActivate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

Виртуальный обработчик внутреннего события "Deactivate" (деактивизация) элемента управления CBmpButton.

```
virtual bool  OnDeactivate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

Виртуальный обработчик события мыши "MouseDown" (нажатие кнопки мыши) элемента управления CBmpButton.

```
virtual bool  OnMouseDown()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

Виртуальный обработчик события мыши "MouseUp" (отпускание кнопки мыши) элемента управления CBmpButton.

```
virtual bool  OnMouseUp()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CButton

Класс CButton является классом простого элемента управления на основе объекта "Кнопка".

Описание

Класс CButton предназначен для создания простейших кнопок.

Декларация

```
class CButton : public CWndObj
```

Заголовок

```
#include <Controls\Button.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndObj
      CButton

Результат работы представленного ниже кода:



Методы класса по группам

Создание	
Create	Создает элемент управления

© 2000-2025, MetaQuotes Ltd.

Создание	
Свойства	
Pressed	Получает/устанавливает значение свойства "Pressed"
Locking	Получает/устанавливает значение свойства "Locking"
Обработка изменения параметров	
OnSetText	Обработчик события "SetText" элемента управления
OnSetColor	Обработчик события "SetColor" элемента управления
OnSetColorBackground	Обработчик события "SetColorBackground" элемента управления
OnSetColorBorder	Обработчик события "SetColorBorder" элемента управления
OnSetFont	Обработчик события "SetFont" элемента управления
OnSetFontSize	Обработчик события "SetFontSize" элемента управления
Обработка внутренних событий	
OnCreate	Обработчик внутреннего события "Create" элемента управления
OnShow	Обработчик внутреннего события "Show" элемента управления
OnHide	Обработчик внутреннего события "Hide" элемента управления
OnMove	Обработчик внутреннего события "Move" элемента управления
OnResize	Обработчик внутреннего события "Resize" элемента управления
OnMouseDown	Обработчик внутреннего события "MouseDown" элемента управления
OnOnMouseUp	Обработчик внутреннего события "MouseUp" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndObj

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Пример создания панели с кнопкой:

```
//+------------------------------------------------------------------+
//|                                                ControlsButton.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 "Панель индикации и диалогов управления. Демонстрация работы кла
#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 allow
#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
```

© 2000-2025, MetaQuotes Ltd.

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!CreateButton2())
      return(false);
   if(!CreateButton3())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| 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                                |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
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;
//--- 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__+" Состояние элемента управления: On");
   else
      Comment(__FUNCTION__+" Состояние элемента управления: Off");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
```

© 2000-2025, 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)
  {
//--- очистим комментарии
   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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CButton.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Pressed (метод Get)

Получает значение свойства "Pressed" (состояние) элемента управления CButton.

```
bool  Pressed()
```

Возвращаемое значение

Свойство "Pressed" элемента управления CButton.

Pressed (метод Set)

Устанавливает значение свойства "Pressed" (состояние) элемента управления CButton.

```
bool  Pressed(
   const bool  pressed      // состояние
   )
```

Параметры

pressed

[in] Новое состояние элемента управления CButton.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Locking (метод Get)

Получает значение свойства "Locking" (фиксация) элемента управления CButton.

```
bool  Locking()
```

Возвращаемое значение

Свойство "Locking" элемента управления CButton.

Locking (метод Set)

Устанавливает значение свойства "Locking" (фиксация) элемента управления CButton.

```
void  Locking(
   const bool  flag      // значение
   )
```

Параметры

flag

[in] Новое значение свойства "Locking" элемента управления CButton.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnSetText

Виртуальный обработчик события "SetText" (изменение свойства OBJPROP_TEXT) элемента управления CButton.

```
virtual bool  OnSetText()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

Виртуальный обработчик события "SetColor" (изменение свойства OBJPROP_COLOR) элемента управления CButton.

```
virtual bool  OnSetColor()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

Виртуальный обработчик события "SetColorBackground" (изменение свойства OBJPROP_BGCOLOR) элемента управления CButton.

```
virtual bool  OnSetColorBackground()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBorder

Виртуальный обработчик события "SetColorBorder" (изменение свойства OBJPROP_BORDER_COLOR) элемента управления CButton.

```
virtual bool  OnSetColorBackground()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

Виртуальный обработчик события "SetFont" (изменение параметра OBJPROP_FONT) элемента управления CButton.

```
virtual bool  OnSetFont()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

Виртуальный обработчик события "SetFontSize" (изменение параметра OBJPROP_FONTSIZE) элемента управления CButton.

```
virtual bool  OnSetFontSize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

Виртуальный обработчик внутреннего события "Create" (создание) элемента управления CButton.

```
virtual bool  OnCreate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnShow

Виртуальный обработчик внутреннего события "Show" (отображение) элемента управления CButton.

```
virtual bool  OnShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) элемента управления CButton.

```
virtual bool  OnHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMove

Виртуальный обработчик внутреннего события "Move" (перемещение) элемента управления CButton.

```
virtual bool  OnMove()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnResize

Виртуальный обработчик внутреннего события "Resize" (изменение размеров) элемента управления CButton.

```
virtual bool  OnResize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

Виртуальный обработчик события мыши "MouseDown" (нажатие кнопки мыши) элемента управления CButton.

```
virtual bool  OnMouseDown()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "MouseDown" происходит при нажатии левой кнопки мыши, если указатель мыши находится на элементе управления.

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

Виртуальный обработчик события мыши "MouseUp" (отпускание кнопки мыши) элемента управления CButton.

```
virtual bool  OnMouseUp()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "MouseUp" происходит при отпускании левой кнопки мыши, если указатель мыши находится на элементе управления.

© 2000-2025, MetaQuotes Ltd.

Класс CEdit

Класс CEdit является классом простого элемента управления на основе объекта "Поле ввода".

Описание

Класс CEdit предназначен для создания редактируемых текстовых полей.

Декларация

```
class CEdit : public CWndObj
```

Заголовок

```
#include <Controls\Edit.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndObj
      CEdit

Результат работы представленного ниже кода:



Методы класса по группам

Создание	
Create	Создает элемент управления

© 2000-2025, MetaQuotes Ltd.

Создание	
Свойства	
ReadOnly	Получает/устанавливает значение свойства "ReadOnly" элемента управления
TextAlign	Получает/устанавливает значение свойства "TextAlign" элемента управления
Обработка событий объекта	
OnObjectEndEdit	Виртуальный обработчик события CHARTEVENT_OBJECT_ENDEDIT объекта графика
Обработка изменения параметров	
OnSetText	Обработчик события "SetText" элемента управления
OnSetColor	Обработчик события "SetColor" элемента управления
OnSetColorBackground	Обработчик события "SetColorBackground" элемента управления
OnSetColorBorder	Обработчик события "SetColorBorder" элемента управления
OnSetFont	Обработчик события "SetFont" элемента управления
OnSetFontSize	Обработчик события "SetFontSize" элемента управления
OnSetZOrder	Обработчик события "SetZOrder" элемента управления
Обработка внутренних событий	
OnCreate	Обработчик внутреннего события "Create" элемента управления
OnShow	Обработчик внутреннего события "Show" элемента управления
OnHide	Обработчик внутреннего события "Hide" элемента управления
OnMove	Обработчик внутреннего события "Move" элемента управления
OnResize	Обработчик внутреннего события "Resize" элемента управления
OnChange	Обработчик внутреннего события "Change" элемента управления
OnClick	Обработчик внутреннего события "Click" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от CWnd

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

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

Методы унаследованные от CWndObj

Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Пример создания панели с редактируемым текстовым полем:

```
//+------------------------------------------------------------------+
//|                                                 ControlsEdit.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 "Панель индикации и диалогов управления. Демонстрация работы кла
#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 allow
#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                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CEdit             m_edit;                          // CEdit объект

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
```

© 2000-2025, MetaQuotes Ltd.

```
   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);
//--- разрешим редактировать содержимое
   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)
  {
//--- очистим комментарии
   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
```

© 2000-2025, MetaQuotes Ltd.

```
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CEdit.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ReadOnly (метод Get)

Получает значение свойства "ReadOnly" (только чтение) элемента управления CEdit.

```
bool  ReadOnly()
```

Возвращаемое значение

Свойство "ReadOnly" элемента управления CEdit.

ReadOnly (метод Set)

Устанавливает новое значение свойства "ReadOnly" (только чтение) элемента управления CEdit.

```
bool  ReadOnly(
   const bool  flag      // значение
   )
```

Параметры

flag

[in] Новое значение свойства "ReadOnly" элемента управления CEdit.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

TextAlign (метод Get)

Получает значение свойства "TextAlign" (тип выравнивания текста по горизонтали) элемента управления CEdit.

```
ENUM_ALIGN_MODE  TextAlign() const
```

Возвращаемое значение

Значение свойства "TextAlign" элемента управления CEdit.

TextAlign (метод Set)

Устанавливает значение свойства "TextAlign" (тип выравнивания текста по горизонтали) элемента управления CEdit.

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // значение свойства
   )
```

Параметры

align

[in] Новое значение свойства "TextAlign".

Возвращаемое значение

true - в случае удачи, false - если не удалось изменить свойство.

© 2000-2025, MetaQuotes Ltd.

OnObjectEndEdit

Виртуальный обработчик события CHARTEVENT_OBJECT_ENDEDIT объекта графика.

```
virtual bool  OnObjectEndEdit()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetText

Виртуальный обработчик события "SetText" (изменение свойства OBJPROP_TEXT) элемента управления CEdit.

```
virtual bool  OnSetText()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

Виртуальный обработчик события "SetColor" (изменение свойства OBJPROP_COLOR) элемента управления CEdit.

```
virtual bool  OnSetColor()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

Виртуальный обработчик события "SetColorBackground" (изменение свойства OBJPROP_BGCOLOR) элемента управления CEdit.

```
virtual bool  OnSetColorBackground()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBorder

Виртуальный обработчик события "SetColorBorder" (изменение свойства OBJPROP_BORDER_COLOR) элемента управления CEdit.

```
virtual bool  OnSetColorBorder()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

Виртуальный обработчик события "SetFont" (изменение свойства OBJPROP_FONT) элемента управления CEdit.

```
virtual bool  OnSetFont()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

Виртуальный обработчик события "SetFontSize" (изменение свойства OBJPROP_FONTSIZE) элемента управления CEdit.

```
virtual bool  OnSetFontSize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

Виртуальный обработчик события "SetZOrder" (изменение свойства OBJPROP_ZORDER) элемента управления CEdit.

```
virtual bool  OnSetZOrder()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

Виртуальный обработчик внутреннего события "Create" (создание) элемента управления CEdit.

```
virtual bool  OnCreate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnShow

Виртуальный обработчик внутреннего события "Show" (отображение) элемента управления CEdit.

```
virtual bool  OnShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) элемента управления CEdit.

```
virtual bool  OnHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMove

Виртуальный обработчик внутреннего события "Move" (перемещение) элемента управления CEdit.

```
virtual bool  OnMove()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnResize

Виртуальный обработчик внутреннего события "Resize" (изменение размеров) элемента управления CEdit.

```
virtual bool  OnResize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChange

Виртуальный обработчик внутреннего события "Change" (изменение) элемента управления CEdit.

```
virtual bool  OnChange()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClick

Виртуальный обработчик внутреннего события "Click" (клик мыши) элемента управления CEdit.

```
virtual bool  OnClick()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CPanel

Класс CPanel является классом простого элемента управления на основе объекта "Прямоугольная метка".

Описание

Класс CPanel предназначен для визуального объединения группы функционально связанных однородных элементов.

Декларация

```
class CPanel : public CWndObj
```

Заголовок

```
#include <Controls\Panel.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndObj
      CPanel

Результат работы представленного ниже кода:



Методы класса по группам

© 2000-2025, MetaQuotes Ltd.

Создание	
Create	Создает элемент управления
Параметры объекта графика	
BorderType	Получает значение параметра "Border" (тип рамки) объекта графика
Обработка событий объекта	
OnSetText	Обработчик события "SetText" элемента управления
OnSetColorBackground	Обработчик события "SetColorBackground" элемента управления
OnSetColorBorder	Обработчик события "SetColorBorder" элемента управления
Обработка внутренних событий	
OnCreate	Обработчик внутреннего события "Create" элемента управления
OnShow	Обработчик внутреннего события "Show" элемента управления
OnHide	Обработчик внутреннего события "Hide" элемента управления
OnMove	Обработчик внутреннего события "Move" элемента управления
OnResize	Обработчик внутреннего события "Resize" элемента управления
OnChange	Обработчик внутреннего события "Change" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Пример создания панели с прямоугольной меткой:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                 ControlsPanel.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 "Панель индикации и диалогов управления. Демонстрация работы кла
#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 allow
#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)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
  }
//+------------------------------------------------------------------+
//| 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(!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;
//---
```

© 2000-2025, MetaQuotes Ltd.

```
CPanel   my_white_border;        // object CPanel
bool     pause=true;             // true - пауза
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
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)
  {
//--- очистим комментарии
   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;
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CPanel.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

BorderType (метод Get)

Получает значение параметра "Border" (тип рамки) объекта графика.

```
ENUM_BORDER_TYPE  BorderType()
```

Возвращаемое значение

Значение параметра "Border".

BorderType (метод Set)

Устанавливает значение параметра "Border" (тип рамки) объекта графика.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // значение
   )
```

Параметры

type

[in] Новое значение параметра "Border".

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetText

Виртуальный обработчик события "SetText" (изменение свойства OBJPROP_TEXT) элемента управления CPanel.

```
virtual bool  OnSetText()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

Виртуальный обработчик события "SetColorBackground" (изменение свойства OBJPROP_BGCOLOR) элемента управления CPanel.

```
virtual bool  OnSetColorBackground()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBorder

Виртуальный обработчик события "SetColorBorder" (изменение свойства OBJPROP_BORDER_COLOR) элемента управления CPanel.

```
virtual bool  OnSetColorBorder()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

Виртуальный обработчик внутреннего события "Create" (создание) элемента управления CPanel.

```
virtual bool  OnCreate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnShow

Виртуальный обработчик внутреннего события "Show" (отображение) элемента управления CPanel.

```
virtual bool  OnShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) элемента управления CPanel.

```
virtual bool  OnHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMove

Виртуальный обработчик внутреннего события "Move" (перемещение) элемента управления CPanel.

```
virtual bool  OnMove()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnResize

Виртуальный обработчик внутреннего события "Resize" (изменение размеров) элемента управления CPanel.

```
virtual bool  OnResize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChange

Виртуальный обработчик внутреннего события "Change" (изменение) элемента управления CPanel.

```
virtual bool  OnChange()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CPicture

Класс CPicture является классом простого элемента управления на основе объекта "Графическая метка".

Описание

Класс CPicture предназначен для создания простых графических изображений.

Декларация

```
class CPicture : public CWndObj
```

Заголовок

```
#include <Controls\Picture.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndObj
      CPicture

Результат работы представленного ниже кода:



Методы класса по группам

Создание	
Create	Создает элемент управления

© 2000-2025, MetaQuotes Ltd.

Создание	
Свойства объекта графика	
Border	Получает/устанавливает ширину рамки объекта графика
BmpName	Получает/устанавливает имя bmp-файла для отображения элемента управления
Обработка внутренних событий	
OnCreate	Обработчик внутреннего события "Create" элемента управления
OnShow	Обработчик внутреннего события "Show" элемента управления
OnHide	Обработчик внутреннего события "Hide" элемента управления
OnMove	Обработчик внутреннего события "Move" элемента управления
OnChange	Обработчик внутреннего события "Change" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Пример создания панели с графической меткой:

```
//+------------------------------------------------------------------+
//|                                                ControlsPicture.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 "Панель индикации и диалогов управления. Демонстрация работы кла
#include <Controls\Dialog.mqh>
#include <Controls\Picture.mqh>
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 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 allow
#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)
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 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(!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);
//--- установим имя bmp-файлов для отображения элемента управления CPicture
   m_picture.BmpName("\\Images\\euro.bmp");

   if(!Add(m_picture))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickPicture(void)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| 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)
  {
//--- очистим комментарии
   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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CPicture.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Border (метод Get)

Получает значение свойства "Border" (ширина рамки) элемента управления CPicture.

```
int  Border()  const
```

Возвращаемое значение

Значение свойства "Border".

Border (метод Set)

Устанавливает значение свойства "Border" (ширина рамки) элемента управления CPicture.

```
bool  Border(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение свойства "Border".

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

BmpName (метод Get)

Получает имя bmp-файла для отображения элемента управления CPicture.

```
string  BmpName()  const
```

Возвращаемое значение

Имя bmp-файла для отображения элемента.

BmpName (метод Set)

Устанавливает имя bmp-файла для отображения элемента управления CPicture.

```
bool  BmpName(
   const string  name      // имя файла
   )
```

Параметры

name

[in] Имя bmp-файла для отображения элемента.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

Виртуальный обработчик внутреннего события "Create" (создание) элемента управления CPicture.

```
virtual bool  OnCreate()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnShow

Виртуальный обработчик внутреннего события "Show" (отображение) элемента управления CPicture.

```
virtual bool  OnShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) элемента управления CPicture.

```
virtual bool  OnHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnMove

Виртуальный обработчик внутреннего события "Move" (перемещение) элемента управления CPicture.

```
virtual bool  OnMove()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChange

Виртуальный обработчик внутреннего события "Change" (изменение) элемента управления CPicture.

```
virtual bool  OnChange()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CScroll

Класс CScroll является базовым классом для создания полос прокрутки.

Описание

Класс CScroll является комбинированным элементом управления, содержащим базовые механизмы для создания полос прокрутки. Базовый класс непосредственно не используется, функциональную нагрузку несут только его наследники - классы CScrollV и CScrollH.

Декларация

```
class CScroll : public CWndContainer
```

Заголовок

```
#include <Controls\Scrolls.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CScroll

Прямые потомки

CScrollH, CScrollV

Методы класса по группам

Создание	
Create	Создает элемент управления
Обработчики событий графика	
OnEvent	Обрабатывает все события графика
Свойства	
MinPos	Получает/устанавливает значение минимальной позиции
MaxPos	Получает/устанавливает значение максимальной позиции
CurrPos	Получает/устанавливает значение текущей позиции
Создание подчиеннных элементов управления	
CreateBack	Создает кнопку-подложку элемента управления
CreateInc	Создает кнопку для увеличения позиции полосы прокрутки

© 2000-2025, MetaQuotes Ltd.

Создание	
CreateDec	Создает кнопку для уменьшения позиции полосы прокрутки
CreateThumb	Создает кнопку-бегунок текущей позиции полосы прокрутки
Обработчики подчиненных элементов управления	
OnClickInc	Обработчик события увеличения позиции полосы прокрутки
OnClickDec	Обработчик события уменьшения позиции полосы прокрутки
Обработка внутренних событий	
OnShow	Обработчик внутреннего события "Create" элемента управления
OnHide	Обработчик внутреннего события "Hide" элемента управления
OnChangePos	Обработчик внутреннего события "ChangePosition" элемента управления
События перемещения объекта	
OnThumbDragStart	Обработчик события "ThumbDragStart" элемента управления
OnThumbDragProcess	Обработчик события "ThumbDragProcess" элемента управления
OnThumbDragEnd	Обработчик события "ThumbDragEnd" элемента управления
Расчет положения по координатам	
CalcPos	Получает позицию полосы прокрутки по координате

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CScroll.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

MinPos (метод Get)

Получает значение параметра "MinPos" (минимальная позиция) элемента управления CScroll.

```
int  MinPos()  const
```

Возвращаемое значение

Значение параметра "MinPos".

MinPos (метод Set)

Устанавливает значение параметра "MinPos" (минимальная позиция) элемента управления CScroll.

```
void  MinPos(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение параметра "MinPos".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

MaxPos (метод Get)

Получает значение параметра "MaxPos" (максимальная позиция) элемента управления CScroll.

```
int  MaxPos()  const
```

Возвращаемое значение

Значение параметра "MaxPos".

MaxPos (метод Set)

Устанавливает значение параметра "MaxPos" (максимальная позиция) элемента управления CScroll.

```
void  MaxPos(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение параметра "MaxPos".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

CurrPos (метод Get)

Получает значение параметра "CurrPos" (текущая позиция) элемента управления CScroll.

```
int  CurrPos()  const
```

Возвращаемое значение

Значение параметра "CurrPos".

CurrPos (метод Set)

Устанавливает значение параметра "CurrPos" (текущая позиция) элемента управления CScroll.

```
void  CurrPos(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение параметра "CurrPos".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

CreateBack

Создает кнопку-подложку элемента управления CScroll.

```
virtual bool  CreateBack()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateInc

Создает кнопку для увеличения позиции полосы прокрутки CScroll.

```
virtual bool  CreateInc()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateDec

Создает кнопку для уменьшения позиции полосы прокрутки CScroll.

```
virtual bool  CreateDec()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateThumb

Создает кнопку-бегунок текущей позиции полосы прокрутки CScroll.

```
virtual bool  CreateThumb()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickInc

Виртуальный обработчик внутреннего события "ClickInc" (клик мыши на ползунок увеличения значения позиции) элемента управления CScroll.

```
virtual bool  OnClickInc()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickDec

Виртуальный обработчик внутреннего события "ClickDec" (клик мыши на ползунок уменьшения значения позиции) элемента управления CScroll.

```
virtual bool  OnClickDec()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnShow

Виртуальный обработчик внутреннего события "Show" (отображение) элемента управления CScroll.

```
virtual bool  OnShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) элемента управления CScroll.

```
virtual bool  OnHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChangePos

Виртуальный обработчик внутреннего события "ChangePos" (изменение позиции) элемента управления CScroll.

```
virtual bool  OnChangePos()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

Виртуальный обработчик события "ThumbDragStart" (начало операции перетаскивания) элемента управления CScroll.

```
virtual bool  OnThumbDragStart()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "ThumbDragStart" происходит при начале операции перетаскивания элемента управления.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

Виртуальный обработчик события "ThumbDragProcess" (операция перетаскивания) элемента управления CScroll.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // координата x
   const int  y       // координата y
   )
```

Параметры

x

[in] Текущее значение координаты X курсора мыши.

y

[in] Текущее значение координаты Y курсора мыши.

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "ThumbDragProcess" происходит при перемещении элемента управления.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragEnd

Виртуальный обработчик события "ThumbDragEnd" (завершение операции перетаскивания) элемента управления CScroll.

```
virtual bool  OnThumbDragEnd()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "ThumbDragEnd" происходит по завершении операции перетаскивания элемента управления.

© 2000-2025, MetaQuotes Ltd.

CalcPos

Получает позицию полосы прокрутки элемента управления CScroll по координате.

```
virtual int  CalcPos(
   const int      coord      // координата
   )
```

Параметры

coord

[in] Координата позиции прокрутки.

Возвращаемое значение

Значение позиции полосы прокрутки.

© 2000-2025, MetaQuotes Ltd.

Класс CScrollV

Класс CScrollV является классом комбинированного элемента управления "Вертикальная полоса прокрутки".

Описание

Класс CScrollV предназначен для создания вертикальных полос прокрутки.

Декларация

```
class CScrollV : public CScroll
```

Заголовок

```
#include <Controls\Scrolls.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CScroll
        CScrollV

Результат работы представленного ниже кода:



Методы класса по группам

© 2000-2025, MetaQuotes Ltd.

Создание подчиеннных элементов управления	
CreateInc	Создает кнопку для увеличения позиции полосы прокрутки
CreateDec	Создает кнопку для уменьшения позиции полосы прокрутки
CreateThumb	Создает кнопку-бегунок текущей позиции полосы прокрутки
Обработка внутренних событий	
OnResize	Обработчик внутреннего события "Resize" элемента управления
OnChangePos	Обработчик внутреннего события "ChangePosition" элемента управления
События перемещения объекта	
OnThumbDragStart	Обработчик события "ThumbDragStart" элемента управления
OnThumbDragProcess	Обработчик события "ThumbDragProcess" элемента управления
OnThumbDragEnd	Обработчик события "ThumbDragEnd" элемента управления
Расчет положения по координатам	
CalcPos	Получает позицию полосы прокрутки по координате

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Методы унаследованные от CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Пример создания панели с вертикальным скроллом:

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollV.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
```

© 2000-2025, 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 "Панель индикации и диалогов управления. Демонстрация работы кла
#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 allow
#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 объект

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
```

© 2000-2025, MetaQuotes Ltd.

```
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| 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
```

© 2000-2025, MetaQuotes Ltd.

```
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
   Comment("Позиция полосы прокрутки ",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Позиция полосы прокрутки ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Позиция полосы прокрутки ",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)
  {
//--- очистим комментарии
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
```

© 2000-2025, 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
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateInc

Создает кнопку для увеличения позиции полосы прокрутки CScrollV.

```
virtual bool  CreateInc()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateDec

Создает кнопку для уменьшения позиции полосы прокрутки CScrollV.

```
virtual bool  CreateDec()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateThumb

Создает кнопку-бегунок текущей позиции полосы прокрутки CScrollV.

```
virtual bool  CreateThumb()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnResize

Виртуальный обработчик внутреннего события "Resize" (изменение размеров) элемента управления CScrollV.

```
virtual bool  OnResize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChangePos

Виртуальный обработчик внутреннего события "ChangePos" (изменение позиции) элемента управления CScrollV.

```
virtual bool  OnChangePos()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

Виртуальный обработчик события "ThumbDragStart" (начало операции перетаскивания) элемента управления CScrollV.

```
virtual bool  OnThumbDragStart()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "ThumbDragStart" происходит при начале операции перетаскивания элемента управления.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

Виртуальный обработчик события "ThumbDragProcess" (операция перетаскивания) элемента управления CScrollV.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // координата x
   const int  y       // координата y
   )
```

Параметры

x

[in] Текущее значение координаты X курсора мыши.

y

[in] Текущее значение координаты Y курсора мыши.

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "ThumbDragProcess" происходит при перемещении элемента управления.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragEnd

Виртуальный обработчик события "ThumbDragEnd" (завершение операции перетаскивания) элемента управления CScrollV.

```
virtual bool  OnThumbDragEnd()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "ThumbDragEnd" происходит по завершении операции перетаскивания элемента управления.

© 2000-2025, MetaQuotes Ltd.

CalcPos

Получает позицию полосы прокрутки элемента управления CScrollV по координате.

```
virtual int  CalcPos(
   const int      coord      // координата
   )
```

Параметры

coord

[in] Координата позиции прокрутки.

Возвращаемое значение

Значение позиции полосы прокрутки.

© 2000-2025, MetaQuotes Ltd.

Класс CScrollH

Класс CScrollH является классом комбинированного элемента управления "Горизонтальная полоса прокрутки".

Описание

Класс CScrollH предназначен для создания горизонтальных полос прокрутки.

Декларация

```
class CScrollH : public CScroll
```

Заголовок

```
#include <Controls\Scrolls.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CScroll
        CScrollH

Результат работы представленного ниже кода:



Методы класса по группам

© 2000-2025, MetaQuotes Ltd.

Создание подчиеннных элементов управления	
CreateInc	Создает кнопку для увеличения позиции полосы прокрутки
CreateDec	Создает кнопку для уменьшения позиции полосы прокрутки
CreateThumb	Создает кнопку-бегунок текущей позиции полосы прокрутки
Обработка внутренних событий	
OnResize	Обработчик внутреннего события "Resize" элемента управления
OnChangePos	Обработчик внутреннего события "ChangePosition" элемента управления
События перемещения объекта	
OnThumbDragStart	Обработчик события "ThumbDragStart" элемента управления
OnThumbDragProcess	Обработчик события "ThumbDragProcess" элемента управления
OnThumbDragEnd	Обработчик события "ThumbDragEnd" элемента управления
Расчет положения по координатам	
CalcPos	Получает позицию полосы прокрутки по координате

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Методы унаследованные от CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Пример создания панели с горизонтальным скроллом:

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollH.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
```

© 2000-2025, 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 "Панель индикации и диалогов управления. Демонстрация работы кла
#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 allow
#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 объект

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
```

© 2000-2025, MetaQuotes Ltd.

```
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| 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
```

© 2000-2025, MetaQuotes Ltd.

```
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
   Comment("Позиция полосы прокрутки ",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Позиция полосы прокрутки ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Позиция полосы прокрутки ",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)
  {
//--- очистим комментарии
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
```

© 2000-2025, 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
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateInc

Создает кнопку для увеличения позиции полосы прокрутки CScrollH.

```
virtual bool  CreateInc()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateDec

Создает кнопку для уменьшения позиции полосы прокрутки CScrollH.

```
virtual bool  CreateDec()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateThumb

Создает кнопку-бегунок текущей позиции полосы прокрутки CScrollH.

```
virtual bool  CreateThumb()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnResize

Виртуальный обработчик внутреннего события "Resize" (изменение размеров) элемента управления CScrollH.

```
virtual bool  OnResize()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChangePos

Виртуальный обработчик внутреннего события "ChangePos" (изменение позиции) элемента управления CScrollH.

```
virtual bool  OnChangePos()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

Виртуальный обработчик события "ThumbDragStart" (начало операции перетаскивания) элемента управления CScrollH.

```
virtual bool  OnThumbDragStart()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "ThumbDragStart" происходит при начале операции перетаскивания элемента управления.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

Виртуальный обработчик события "ThumbDragProcess" (операция перетаскивания) элемента управления CScrollH.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // координата x
   const int  y       // координата y
   )
```

Параметры

x

[in] Текущее значение координаты X курсора мыши.

y

[in] Текущее значение координаты Y курсора мыши.

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "ThumbDragProcess" происходит при перемещении элемента управления.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragEnd

Виртуальный обработчик события "ThumbDragEnd" (завершение операции перетаскивания) элемента управления CScrollH.

```
virtual bool  OnThumbDragEnd()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "ThumbDragEnd" происходит по завершении операции перетаскивания элемента управления.

© 2000-2025, MetaQuotes Ltd.

CalcPos

Получает позицию полосы прокрутки элемента управления CScrollH по координате.

```
virtual int  CalcPos(
   const int      coord      // координата
   )
```

Параметры

coord

[in] Координата позиции прокрутки.

Возвращаемое значение

Значение позиции полосы прокрутки.

© 2000-2025, MetaQuotes Ltd.

Класс CWndClient

Класс CWndClient представляет собой комбинированный элемент управления "Клиентская область" и является базовым классом для создания областей с полосами прокрутки.

Описание

Класс CWndClient является программной реализацией функционала для создания областей с полосами прокрутки.

Декларация

```
class CWndClient : public CWndContainer
```

Заголовок

```
#include <Controls\WndClient.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CWndClient

Прямые потомки

CCheckGroup, CListView, CRadioGroup

Методы класса по группам

Создание	
Create	Создает элемент управления
Обработка событий графика	
OnEvent	Обрабатывает все события графика
Параметры	
ColorBackground	Устанавливает цвет фона элемента
ColorBorder	Устанавливает цвет рамки элемента
BorderType	Устанавливает тип рамки элемента
Настройки	
VScrolled	Получает/устанавливает признак использования вертикальной полосы прокрутки
HScrolled	Получает/устанавливает признак использования горизонтальной полосы прокрутки

© 2000-2025, MetaQuotes Ltd.

Создание	
Подчиненные элементы управления	
CreateBack	Создает кнопку-подложку полосы прокрутки
CreateScrollV	Создает вертикальную полосу прокрутки
CreateScrollH	Создает горизонтальную полосу прокрутки
Обработка внутренних событий	
OnResize	Виртуальный обработчик внутреннего события "Resize" элемента управления
Обработка событий подчиненных элементов управления	
OnVScrollShow	Виртуальный обработчик внутреннего события "Show" подчиненного элемента управления VScroll
OnVScrollHide	Виртуальный обработчик внутреннего события "Hide" подчиненного элемента управления VScroll
OnHScrollShow	Виртуальный обработчик внутреннего события "Show" подчиненного элемента управления HScroll
OnHScrollHide	Виртуальный обработчик внутреннего события "Hide" подчиненного элемента управления HScroll
OnScrollLineDown	Виртуальный обработчик внутреннего события "ScrollLineDown" подчиненного элемента управления VScroll
OnScrollLineUp	Виртуальный обработчик внутреннего события "ScrollLineUp" подчиненного элемента управления VScroll
OnScrollLineLeft	Виртуальный обработчик внутреннего события "ScrollLineLeft" подчиненного элемента управления HScroll
OnScrollLineRight	Виртуальный обработчик внутреннего события "ScrollLineRight" подчиненного элемента управления HScroll
Изменение размеров	
Rebound	Устанавливает новые параметры области элемента управления из координат класса CRect

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Методы унаследованные от CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CWndClient.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ColorBackground

Устанавливает цвет фона элемента управления.

```
bool  ColorBackground(
   const color  value      // цвет фона
   )
```

Параметры

value

[in] Цвет фона элемента управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ColorBorder

Устанавливает цвет рамки элемента управления.

```
bool  ColorBorder(
   const color  value      // цвет
   )
```

Параметры

value

[in] Цвет рамки элемента управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

BorderType

Устанавливает тип рамки элемента управления.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // значение
   )
```

Параметры

type

[in] Тип рамки элемента управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

VScrolled (метод Get)

Получает признак использования вертикальной полосы прокрутки.

```
bool  VScrolled()
```

Возвращаемое значение

true, если вертикальная полоса прокрутки используется, иначе - false.

VScrolled (метод Set)

Устанавливает признак использования вертикальной полосы прокрутки.

```
bool  VScrolled(
   const bool  flag      // флаг
   )
```

Параметры

flag

[in] Флаг.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

HScrolled (метод Get)

Получает признак использования горизонтальной полосы прокрутки.

```
bool  HScrolled()
```

Возвращаемое значение

true, если горизонтальная полоса прокрутки используется, иначе - false.

HScrolled (метод Set)

Устанавливает признак использования горизонтальной полосы прокрутки.

```
bool  HScrolled(
   const bool  flag      // флаг
   )
```

Параметры

flag

[in] Флаг.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateBack

Создает кнопку-подложку полосы прокрутки.

```
virtual bool  CreateBack()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateScrollV

Создает вертикальную полосу прокрутки.

```
virtual bool  CreateScrollV()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateScrollH

Создает горизональную полосу прокрутки.

```
virtual bool  CreateScrollH()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnResize

Виртуальный обработчик внутреннего события "Resize" (изменение размеров) элемента управления CWinClient.

```
virtual bool  OnResize()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

Виртуальный обработчик внутреннего события "Show" (отображение) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnVScrollShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnVScrollHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnHScrollShow

Виртуальный обработчик внутреннего события "Show" (отображение) подчиненного элемента управления HScroll (горизонтальная полоса прокрутки).

```
virtual bool  OnHScrollShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnHScrollHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) подчиненного элемента управления HScroll (горизонтальная полоса прокрутки).

```
virtual bool  OnHScrollHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

Виртуальный обработчик внутреннего события "ScrollLineDown" (прокрутка на строку вниз) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnScrollLineDown()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

Виртуальный обработчик внутреннего события "ScrollLineUp" (прокрутка на строку вверх) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnScrollLineUp()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineLeft

Виртуальный обработчик внутреннего события "ScrollLineLeft" (прокрутка влево) подчиненного элемента управления HScroll (горизонтальная полоса прокрутки).

```
virtual bool  OnScrollLineLeft()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineRight

Виртуальный обработчик внутреннего события "ScrollLineRight" (прокрутка вправо) подчиненного элемента управления HScroll (горизонтальная полоса прокрутки).

```
virtual bool  OnScrollLineRight()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Метод базового класса не выполняет никаких действий и всегда возвращает true.

© 2000-2025, MetaQuotes Ltd.

ReBound

Устанавливает новые параметры области элемента управления CWndClient из координат класса CRect.

```
void  ReBound(
   const & CRect  rect      // класс прямоугольной области
   )
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Класс CListView

Класс CListView является классом комбинированного элемента управления "Просмотр списка".

Описание

Класс CListView предназначен для отображения списка элементов данных.

Декларация

```
class CListView : public CWndClient
```

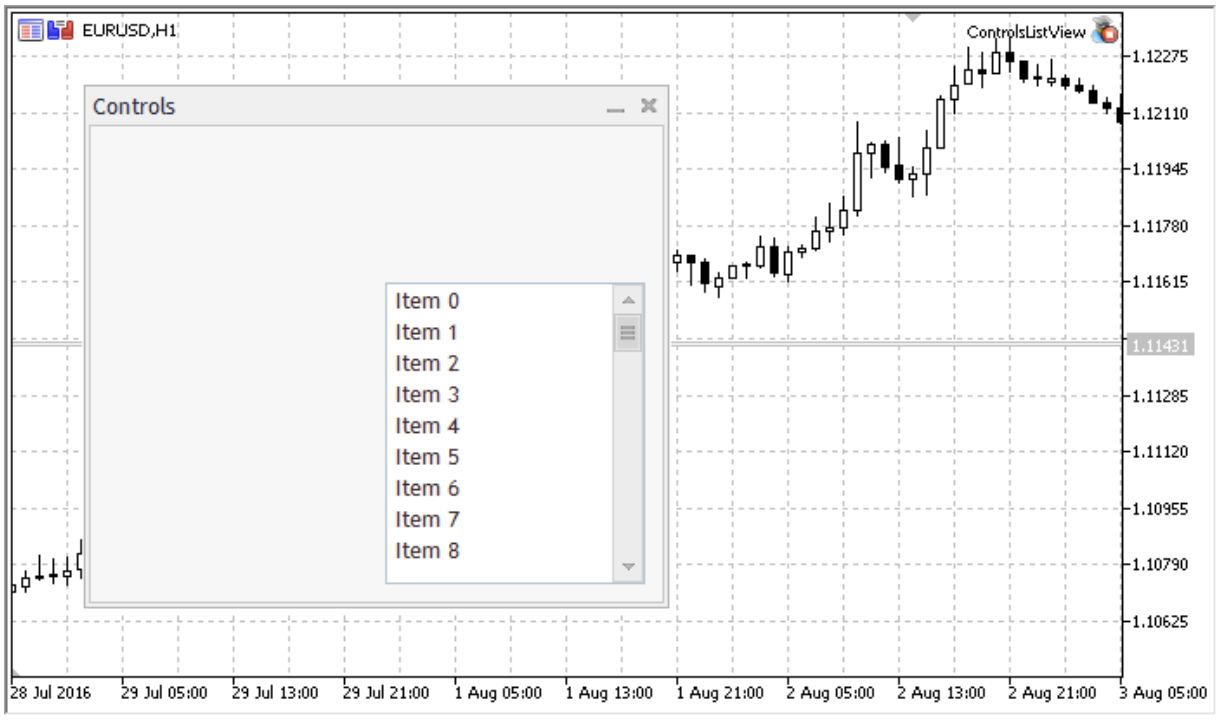
Заголовок

```
#include <Controls\ListView.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CWndClient
        CListView

Результат работы представленного ниже кода:



Методы класса по группам

© 2000-2025, MetaQuotes Ltd.

Создание	
Create	Создает элемент управления
Обработка событий графика	
OnEvent	Обрабатывает все события графика
Настройка	
TotalView	Установка параметра, определяющего количество отображаемых элементов списка
Наполнение	
AddItem	Добавляет элемент (строку) в список элемента управления
Данные	
Select	Устанавливает текущий элемент списка по индексу
SelectByText	Устанавливает текущий элемент списка по тексту
SelectByValue	Устанавливает текущий элемент списка по значению
Данные (только чтение)	
Value	Получает значение текущего элемента списка
Подчиненные элементы управления	
CreateRow	Создает "строку" элемента списка
Обработка внутренних событий	
OnResize	Виртуальный обработчик внутреннего события "Resize" элемента управления
Обработка событий подчиненных элементов управления	
OnVScrollShow	Виртуальный обработчик внутреннего события "Show" подчиненного элемента управления VScroll
OnVScrollHide	Виртуальный обработчик внутреннего события "Hide" подчиненного элемента управления VScroll
OnScrollLineDown	Виртуальный обработчик внутреннего события "ScrollLineDown" подчиненного элемента управления VScroll
OnScrollLineUp	Виртуальный обработчик внутреннего события "ScrollLineUp" подчиненного элемента управления VScroll
OnItemClick	Виртуальный обработчик внутреннего события "ItemClick" на заданной строке элемента управления

© 2000-2025, MetaQuotes Ltd.

Создание	
Перерисовка	
Redraw	Осуществляет перерисовку элемента управления
RowState	Осуществляет изменение состояния заданной строки элемента управления
CheckView	Осуществляет проверку "видимости" выбранной строки элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

Методы унаследованные от CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Пример создания панели со списком значений:

```
//+------------------------------------------------------------------+
//|                                                ControlsListView.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 "Панель индикации и диалогов управления. Демонстрация работы кла
#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 allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
```

© 2000-2025, MetaQuotes Ltd.

```
#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 объект

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                                                       |
```

© 2000-2025, 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(!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()+"\"");
  }
//+------------------------------------------------------------------+
```

© 2000-2025, 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)
  {
//--- очистим комментарии
   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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CListView.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

TotalView

Устанавливает количество видимых элементов списка элемента управления CListView.

```
bool  TotalView(
   const int  value      // количество видимых элементов
   )
```

Параметры

value

[in] Количество видимых элементов списка.

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Количество отображаемых элементов списка можно установить только один раз.

© 2000-2025, MetaQuotes Ltd.

AddItem

Добавляет элемент (строку) в список элемента управления CListView.

```
bool  AddItem(
   const string  item,      // текст
   const long    value      // значение
   )
```

Параметры

item

[in] Текст.

value

[in] Значение.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Select

Устанавливает текущий элемент списка по индексу.

```
bool  Select(
   const int  index     // индекс
   )
```

Параметры

index

[in] Индекс элемента.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

SelectByText

Устанавливает текущий элемент списка по тексту.

```
bool  SelectByText(
   const string  text     // текст
   )
```

Параметры

text

[in] Текст элемента.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

SelectByValue

Устанавливает текущий элемент списка по значению.

```
bool  SelectByValue(
   const long  value     // значение
   )
```

Параметры

value

[in] Значение.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Value

Получает значение текущего элемента списка.

```
long  Value()
```

Возвращаемое значение

Значение текущего элемента списка.

© 2000-2025, MetaQuotes Ltd.

CreateRow

Создает "строку" элемента списка.

```
bool  CreateRow(
   const int  index     // индекс
   )
```

Параметры

index

[in] Индекс элемента.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnResize

Виртуальный обработчик внутреннего события "Resize" (изменение размеров) элемента управления CListView.

```
virtual bool  OnResize()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

Виртуальный обработчик внутреннего события "Show" (отображение) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnVScrollShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnVScrollHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

Виртуальный обработчик внутреннего события "ScrollLineDown" (прокрутка на строку вниз) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnScrollLineDown()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

Виртуальный обработчик внутреннего события "ScrollLineUp" (прокрутка на строку вверх) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnScrollLineUp()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnItemClick

Виртуальный обработчик внутреннего события "ItemClick" (клик мыши) на заданной строке элемента управления CListView.

```
virtual bool  OnItemClick()
   const int    index     // индекс
   )
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Redraw

Осуществляет перерисовку элемента управления CListView.

```
bool  Redraw()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

RowState

Осуществляет изменение состояния заданной строки элемента управления CListView.

```
bool  RowState(
   const int   index      // индекс
   const bool  select     // состояние
   )
```

Параметры

index

[in] Индекс строки.

select

[in] Состояние строки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CheckView

Осуществляет проверку "видимости" выбранной строки элемента управления CListView.

```
bool  CheckView()
```

Возвращаемое значение

true, если выделенная строка отображается, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CComboBox

Класс CComboBox является классом комбинированного элемента управления "Поле с выпадающим списком".

Описание

Класс CComboBox представляет собой элемент управления с раскрывающимся списком, предназначенный для выбора значения.

Декларация

```
class CComboBox : public CWndContainer
```

Заголовок

```
#include <Controls\ComboBox.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CComboBox

Результат работы представленного ниже кода:



Методы класса по группам

© 2000-2025, MetaQuotes Ltd.

Создание	
Create	Создает элемент управления
Обработка событий графика	
OnEvent	Обрабатывает все события графика
Наполнение	
AddItem	Добавляет элемент (строку) в список элемента управления
Настройка	
ListViewItems	Устанавливает количество элементов выпадающего списка элемента управления
Данные	
Select	Устанавливает текущий элемент списка по индексу
SelectByText	Устанавливает текущий элемент списка по тексту
SelectByValue	Устанавливает текущий элемент списка по значению
Данные (только чтение)	
Value	Получает значение текущего элемента списка
Подчиненные элементы управления	
CreateEdit	Создает подчиненный элемент управления (строка ввода) элемента управления
CreateButton	Создает подчиненный элемент управления (кнопка) элемента управления
CreateList	Создает подчиненный элемент управления (выпадающий список) элемента управления
Обработка событий подчиненных элементов управления	
OnClickEdit	Виртуальный обработчик внутреннего события "ClickEdit" элемента управления
OnClickButton	Виртуальный обработчик внутреннего события "ClickButton" элемента управления
OnChangeList	Виртуальный обработчик внутреннего события "ChangeList" элемента управления
Отображение выпадающего списка	
ListShow	Отображает выпадающий список элемента управления

© 2000-2025, MetaQuotes Ltd.

Создание	
ListHide	Скрывает выпадающий список элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Hide

Пример создания панели с элементом управления "Поле с выпадающим списком":

```
//+------------------------------------------------------------------+
//|                                                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 allow
#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
```

© 2000-2025, MetaQuotes Ltd.

```
#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))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- create dependent controls
   if(!CreateComboBox())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| 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))
```

© 2000-2025, MetaQuotes Ltd.

```
      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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CComboBox.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

AddItem

Добавляет элемент (строку) в список элемента управления CComboBox.

```
bool  AddItem(
   const string  item,      // текст
   const long    value      // значение
   )
```

Параметры

item

[in] Текст.

value=0

[in] Значение.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ListViewItems

Устанавливает количество элементов выпадающего списка элемента управления CComboBox.

```
void  ListViewItems(
   const int     value      // количество элементов
   )
```

Параметры

value

[in] Количество элементов выпадающего списка.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Select

Устанавливает текущий элемент списка по индексу.

```
bool  Select(
   const int  index     // индекс
   )
```

Параметры

index

[in] Индекс элемента.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

SelectByText

Устанавливает текущий элемент списка по указанному тексту.

```
bool  SelectByText(
   const string  text     // текст
   )
```

Параметры

text

[in] Текст элемента.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

SelectByValue

Устанавливает текущий элемент списка по указанному значению.

```
bool  SelectByValue(
   const long  value     // значение
   )
```

Параметры

value

[in] Значение.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Value

Получает значение текущего элемента списка.

```
long  Value()
```

Возвращаемое значение

Значение текущего элемента списка.

© 2000-2025, MetaQuotes Ltd.

CreateEdit

Создает подчиненный элемент управления (строка ввода) элемента управления CComboBox.

```
virtual bool  CreateEdit()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Создает подчиненный элемент управления (кнопка) элемента управления CComboBox.

```
virtual bool  CreateButton()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateList

Создает подчиненный элемент управления (выпадающий список) элемента управления CComboBox.

```
virtual bool  CreateList()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickEdit

Виртуальный обработчик внутреннего события "ClickEdit" (клик мыши на строке ввода) элемента управления CComboBox.

```
virtual bool  OnClickEdit()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickButton

Виртуальный обработчик внутреннего события "ClickButton" (клик мыши на кнопке) элемента управления CComboBox.

```
virtual bool  OnClickButton()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChangeList

Виртуальный обработчик внутреннего события "ChangeList" (изменение выпадающего списка) элемента управления CComboBox.

```
virtual bool  OnChangeList()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ListShow

Отображает выпадающий списокок элемента управления CComboBox.

```
virtual bool  ListShow()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ListHide

Скрывает выпадающий списокк элемента управления CComboBox.

```
virtual bool  ListHide()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CCheckBox

Класс CCheckBox является классом комбинированного элемента управления "Переключатель с фиксацией".

Описание

Класс CCheckBox предназначен для создания элемента управления, изменяющего свое состояние на противоположное при клике мыши.

Декларация

```
class CCheckBox : public CWndContainer
```

Заголовок

```
#include <Controls\CheckBox.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CCheckBox

Результат работы представленного ниже кода:



Методы класса по группам

© 2000-2025, MetaQuotes Ltd.

Создание	
Create	Создает элемент управления
Обработка событий графика	
OnEvent	Обрабатывает все события графика
Настройки	
Text	Получает/устанавливает текст поясняющей надписи элемента управления
Color	Получает/устанавливает цвет текста поясняющей надписи элемента управления
Состояние	
Checked	Получает/устанавливает состояние элемента управления
Данные	
Value	Получает/устанавливает значение, ассоциированное с элементом управления
Подчиненные элементы управления	
CreateButton	Создает подчиненный элемент управления (кнопка) элемента управления
CreateLabel	Создает подчиненный элемент управления (метка поясняющей надписи) элемента управления
Обработка событий подчиненных элементов	
ClickButton	Виртуальный обработчик внутреннего события "ClickButton" элемента управления
ClickLabel	Виртуальный обработчик внутреннего события "ClickLabel" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndContainer

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Пример создания панели с элементом управления "Переключатель с фиксацией":

```
//+------------------------------------------------------------------+
//|                                              ControlsCheckBox.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 "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 allow
#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:
```

© 2000-2025, MetaQuotes Ltd.

```
                     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);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 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;
   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);
```

© 2000-2025, MetaQuotes Ltd.

```
   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()));
  }
//+------------------------------------------------------------------+
//| 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
```

© 2000-2025, MetaQuotes Ltd.

```
                  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);
}
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CCheckBox.

```
virtual bool  Create(
   const long    chart,      // идентификатор графика
   const string  name,       // имя
   const int     subwin,     // подокно графика
   const int     x1,         // координата
   const int     y1,         // координата
   const int     x2,         // координата
   const int     y2          // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Text (метод Get)

Получает текст поясняющей надписи элемента управления CCheckBox.

```
string  Text()
```

Возвращаемое значение

Текст надписи.

Text (метод Set)

Устанавливает текст поясняющей надписи элемента управления CCheckBox.

```
bool  Text(
   const string  value      // значение
   )
```

Параметры

value

[in] Новый текст надписи.

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Текст поясняющей надписи задается путем установки значения свойства OBJPROP_TEXT (текст) объекта графика.

© 2000-2025, MetaQuotes Ltd.

Color (метод Get)

Получает цвет текста поясняющей надписи элемента управления CCheckBox.

```
color  Color()  const
```

Возвращаемое значение

Цвет текста поясняющей надписи.

Color (метод Set)

Устанавливает цвет поясняющей надписи элемента управления CCheckBox.

```
bool  Color(
   const color  value      // цвет
   )
```

Параметры

value

[in] Новый цвет поясняющей надписи.

Возвращаемое значение

true - в случае удачи, иначе - false.

Примечание

Цвет текста поясняющей надписи задается путем установки значения свойства OBJPROP_COLOR (цвет) объекта графика.

© 2000-2025, MetaQuotes Ltd.

Checked (метод Get)

Получает состояние элемента управления CCheckBox.

```
bool  Checked()  const
```

Возвращаемое значение

Состояние элемента управления CCheckBox.

Checked (метод Set)

Устанавливает состояние элемента управления CCheckBox.

```
bool  Checked(
   const bool  flag      // состояние
   )
```

Параметры

flag

[in] Новое состояние.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Value (метод Get)

Получает значение, ассоциированное с элементом управления CCheckBox.

```
int  Value()  const
```

Возвращаемое значение

Значение, ассоциированное с элементом управления CCheckBox.

Value (метод Set)

Устанавливает значение, ассоциированное с элементом управления CCheckBox.

```
void  Value(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Создает подчиненный элемент управления (кнопка) элемента управления CCheckBox.

```
virtual bool  CreateButton()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateLabel

Создает подчиненный элемент управления (метка поясняющей надписи) элемента управления CComboBox.

```
virtual bool  CreateLabel()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickButton

Виртуальный обработчик внутреннего события "ClickButton" (клик мыши на кнопке) элемента управления CCheckBox.

```
virtual bool  OnClickButton()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickLabel

Виртуальный обработчик внутреннего события "ClickLabel" (клик мыши на надписи) элемента управления CCheckBox.

```
virtual bool  OnClickLabel()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CCheckGroup

Класс CCheckGroup является классом комбинированного элемента управления "Переключатель с независимой фиксацией".

Описание

Класс CCheckGroup предназначен для создания элемента управления, позволяющего отображать и редактировать набор флагов.

Декларация

```
class CCheckGroup : public CWndClient
```

Заголовок

```
#include <Controls\CheckGroup.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CWndClient
        CCheckGroup

Результат работы представленного ниже кода:



Методы класса по группам

© 2000-2025, MetaQuotes Ltd.

Создание	
Create	Создает элемент управления
Обработка событий графика	
OnEvent	Обрабатывает все события графика
Наполнение	
AddItem	Добавляет новый элемент в группу
Данные (только чтение)	
Value	Получает значение, ассоциированное с элементом управления
Подчиненные элементы управления	
CreateButton	Создает новый элемент CCheckBox в группе по указанному индексу
Обработка событий подчиненных элементов	
OnVScrollShow	Виртуальный обработчик внутреннего события "Show" подчиненного элемента VScroll
OnVScrollHide	Виртуальный обработчик внутреннего события "Hide" подчиненного элемента VScroll
OnScrollLineDown	Виртуальный обработчик внутреннего события "ScrollLineUp" подчиненного элемента VScroll
OnScrollLineUp	Виртуальный обработчик внутреннего события "ScrollLineDown" подчиненного элемента VScroll
OnChangeItem	Виртуальный обработчик внутреннего события "ChangeItem" элемента управления
Перерисовка	
Redraw	Перерисовывает группу элементов
RowState	Изменяет состояние элемента группы

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Методы унаследованные от CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Пример создания панели с элементом управления "Переключатель с независимой фиксацией":

```
//+------------------------------------------------------------------+
//|                                            ControlsCheckGroup.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 "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 allow
#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                   |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
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);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 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);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CCheckGroup.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

AddItem

Добавляет в группу новый элемент (строку).

```
virtual bool  AddItem(
   const string  item,         // текст
   const long    value=0       // значение
   )
```

Параметры

item

[in] Текст поясняющей надписи добавляемого элемента управления.

value=0

[in] Значение, ассоциированное с добавляемым элементом управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Value

Получает значение, ассоциированное с элементом управления.

```
long  Value()
```

Возвращаемое значение

Значение, ассоциированное с элементом управления.

Примечание

Значение зависит от состояний всех элементов, входящих в группу CCheckGroup.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Создает новый элемент CCheckBox в группе по указанному индексу.

```
bool  CreateButton(
   int  index      // индекс
   )
```

Параметры

index

[in] Индекс нового элемента CheckBox в группе.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

Виртуальный обработчик внутреннего события "Show" (отображение) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnVScrollShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnVScrollHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

Виртуальный обработчик внутреннего события "ScrollLineDown" (прокрутка на одну позицию вниз) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnScrollLineDown()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

Виртуальный обработчик внутреннего события "ScrollLineUp" (прокрутка на одну позицию вверх) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnScrollLineUp()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChangeItem

Виртуальный обработчик внутреннего события "ChangeItem" (изменение состояния элемента группы) элемента управления CCheckGroup.

```
virtual bool  OnChangeItem(
   const int  index      // индекс
   )
```

Параметры

index

[in] Индекс элемента, состояние которого изменилось.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Redraw

Перерисовывает группу элементов.

```
bool  Redraw()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

RowState

Изменяет состояние элемента группы.

```
bool  RowState(
   const int   index,      // индекс
   const bool  select      // состояние
   )
```

Параметры

index

[in] Индекс элемента, состояние которого изменяется.

select

[in] Новое состояние элемента.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CRadioButton

Класс CRadioButton является классом комбинированного элемента управления "Переключатель".

Описание

Класс CRadioButton самостоятельного применения не имеет и используется для создания переключателей с зависимой фиксацией.

Декларация

```
class CRadioButton : public CWndContainer
```

Заголовок

```
#include <Controls\RadioButton.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CRadioButton

Методы класса по группам

Создание	
Create	Создает элемент управления
Обработка событий графика	
OnEvent	Обрабатывает все события графика
Настройки	
Text	Получает/устанавливает текст надписи, связанной с элементом управления
Color	Получает/устанавливает цвет текста поясняющей надписи элемента управления
Состояние	
State	Получает/устанавливает состояние элемента управления
Подчиненные элементы управления	
CreateButton	Создает кнопку
CreateLabel	Создает поясняющую надпись к кнопке

© 2000-2025, MetaQuotes Ltd.

Создание	
Обработка событий подчиненных элементов управления	
OnClickButton	Виртуальный обработчик внутреннего события "ClickButton" элемента управления
OnClickLabel	Виртуальный обработчик внутреннего события "ClickLabel" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CRadioButton.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Text (метод Get)

Получает текст надписи, связанной с элементом управления CRadioButton.

```
string  Text()  const
```

Возвращаемое значение

Текст надписи.

Text (метод Set)

Устанавливает текст надписи.

```
bool  Text(
   const string  value      // текст
   )
```

Параметры

value

[in] Новый текст надписи.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Color (метод Get)

Получает цвет текста поясняющей надписи элемента управления CRadioButton.

```
color  Color()  const
```

Возвращаемое значение

Цвет текста поясняющей надписи.

Color (метод Set)

Устанавливает цвет поясняющей надписи элемента управления CRadioButton.

```
bool  Color(
   const color  value      // цвет
   )
```

Параметры

value

[in] Новый цвет поясняющей надписи.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

State (метод Get)

Получает состояние элемента управления CRadioButton.

```
bool  State()  const
```

Возвращаемое значение

Состояние кнопки.

State (метод Set)

Устанавливает состояние элемента управления CRadioButton.

```
bool  State(
   const bool  flag      // состояние
   )
```

Параметры

flag

[in] Новое состояние кнопки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Создает кнопку.

```
virtual bool  CreateButton()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateLabel

Создает поясняющую надпись к кнопке.

```
virtual bool  CreateLabel()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickButton

Виртуальный обработчик внутреннего события "ClickButton" (клик кнопки) элемента управления CRadioButton.

```
virtual bool  OnClickButton()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickLabel

Виртуальный обработчик внутреннего события "ClickLabel" (клик надписи) элемента управления CRadioButton.

```
virtual bool  OnClickLabel()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CRadioGroup

Класс CRadioGroup является классом комбинированного элемента управления "Переключатель с зависимой фиксацией".

Описание

Класс CRadioGroup предназначен для создания элемента управления, позволяющего отображать и редактировать поле перечислимого типа.

Декларация

```
class CRadioGroup : public CWndClient
```

Заголовок

```
#include <Controls\RadioGroup.mqh>
```

Иерархия наследования

CObject
CWnd
CWndContainer
CWndClient
CRadioGroup

Результат работы представленного ниже кода:



Методы класса по группам

© 2000-2025, MetaQuotes Ltd.

Создание	
Create	Создает элемент управления
Обработка событий графика	
OnEvent	Обрабатывает все события графика
Наполнение	
AddItem	Добавляет в группу новый элемент
Данные (только чтение)	
Value	Получает значение, ассоциированное с состоянием элемента управления
Подчиненные элементы управления	
CreateButton	Создает новый элемент в группе по указанному индексу
Обработка событий подчиненных элементов управления	
OnVScrollShow	Виртуальный обработчик внутреннего события "Show" подчиненного элемента VScroll
OnVScrollHide	Виртуальный обработчик внутреннего события "Hide" подчиненного элемента VScroll
OnScrollLineDown	Виртуальный обработчик внутреннего события "ScrollLineDown" подчиненного управления VScroll
OnScrollLineUp	Виртуальный обработчик внутреннего события "ScrollLineUp" подчиненного элемента VScroll
OnChangeItem	Виртуальный обработчик внутреннего события "ChangeItem" элемента управления
Перерисовка	
Redraw	Перерисовывает группу элементов
RowState	Изменяет состояние элемента группы
Select	Выбирает текущий элемент

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Методы унаследованные от CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Методы унаследованные от CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Пример создания панели с группой "радиокнопок":

```
//+------------------------------------------------------------------+
//|                                          ControlsRadioGroup.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 "Панель индикации и диалогов управления. Демонстрация работы кла
#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 allow
#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                   |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CRadioGroup       m_radio_group;                    // CRadioGroup объект

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
```

© 2000-2025, MetaQuotes Ltd.

```
   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
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- очистим комментарии
   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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CRadioGroup.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

AddItem

Добавляет в группу новый элемент.

```
virtual bool  AddItem(
   const string  item,         // текст
   const long    value=0       // значение
   )
```

Параметры

item

[in] Текст поясняющей надписи добавляемого элемента управления.

value=0

[in] Значение, ассоциированное с добавляемым элементом управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Value

Получает значение, ассоциированное с состоянием элемента управления.

```
long  Value()
```

Возвращаемое значение

Значение, ассоциированное с состоянием элемента управления.

Примечание

Значение зависит от состояний всех элементов CRadioButton, входящих в группу.

© 2000-2025, MetaQuotes Ltd.

CreateButton

Создает новый элемент в группе по указанному индексу.

```
bool  CreateButton(
   const int  index      // индекс
   )
```

Параметры

index

[in] Индекс нового элемента в группе.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

Виртуальный обработчик внутреннего события "Show" (отображение) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnVScrollShow()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

Виртуальный обработчик внутреннего события "Hide" (сокрытие) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnVScrollHide()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

Виртуальный обработчик внутреннего события "ScrollLineDown" (прокрутка на одну позицию вниз) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnScrollLineDown()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

Виртуальный обработчик внутреннего события "ScrollLineUp" (прокрутка на одну позицию вверх) подчиненного элемента управления VScroll (вертикальная полоса прокрутки).

```
virtual bool  OnScrollLineUp()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChangeItem

Виртуальный обработчик внутреннего события "ChangeItem" (изменение состояния элемента группы) элемента управления CRadioGroup.

```
virtual bool  OnChangeItem(
   const int  index      // индекс
   )
```

Параметры

index

[in] Индекс элемента, состояние которого изменилось.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Redraw

Перерисовывает группу элементов.

```
bool  Redraw()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

RowState

Изменяет состояние элемента группы.

```
bool  RowState(
   const int   index,      // индекс
   const bool  select      // состояние
   )
```

Параметры

index

[in] Индекс элемента, состояние которого изменяется.

select

[in] Новое состояние элемента.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Select

Выбирает текущий элемент.

```
void  Select(
   const int  index      // индекс
   )
```

Параметры

index

[in] Индекс выбираемого элемента.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Класс CSpinEdit

Класс CSpinEdit является классом комбинированного элемента управления "Поле инкремента-декремента".

Описание

Класс CSpinEdit позволяет создавать элемент управления для редактирования значения целочисленной переменной с указанным шагом в указанных пределах.

Декларация

```
class CSpinEdit : public CWndContainer
```

Заголовок

```
#include <Controls\SpinEdit.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CSpinEdit

Результат работы представленного ниже кода:



Методы класса по группам

© 2000-2025, MetaQuotes Ltd.

Создание	
Create	Создает элемент управления
Обработчик событий графика	
OnEvent	Обрабатывает все события графика
Настройка	
MinValue	Получает/устанавливает минимальное значение элемента управления
MaxValue	Получает/устанавливает максимальное значение элемента управления
Состояние	
Value	Получает/устанавливает текущее значение элемента управления
Подчиненные элементы управления	
CreateEdit	Создает подчиненный элемент управления CEdit элемента управления
CreateInc	Создает кнопку для увеличения значения (инкремент) элемента управления
CreateDec	Создает кнопку для уменьшения значения (декремент) элемента управления
Обработка событий подчиненных элементов управления	
OnClickInc	Виртуальный обработчик внутреннего события "ClickInc" элемента управления
OnClickDec	Виртуальный обработчик внутреннего события "ClickDec" элемента управления
Обработка внутренних событий	
OnChangeValue	Виртуальный обработчик внутреннего события "ChangeValue" элемента управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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,

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Методы унаследованные от CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Пример создания панели с прокруткой значений:

```
//+------------------------------------------------------------------+
//|                                              ControlsSpinEdit.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 "Панель индикации и диалогов управления. Демонстрация работы кла
#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 allow
#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
```

© 2000-2025, MetaQuotes Ltd.

```
  {
private:
   CSpinEdit         m_spin_edit;                     // CSpinEdit объект

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);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 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))
      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                                 |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnDeinit(const int reason)
  {
//--- очистим комментарии
   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);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CSpinEdit.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

MinValue (метод Get)

Получает значение параметра "MinValue" (минимальное значение) элемента управления CSpinEdit.

```
int  MinValue()  const
```

Возвращаемое значение

Значение параметра "MinValue".

MinValue (метод Set)

Устанавливает значение параметра "MinValue" (минимальное значение) элемента управления CSpinEdit.

```
void  MinValue(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение параметра "MinValue".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

MaxValue (метод Get)

Получает значение параметра "MaxValue" (максимальное значение) элемента управления CSpinEdit.

```
int  MaxValue()  const
```

Возвращаемое значение

Значение параметра "MaxValue".

MaxValue (метод Set)

Устанавливает значение параметра "MaxValue" (максимальное значение) элемента управления CSpinEdit.

```
void  MaxValue(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение параметра "MaxValue".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Value (метод Get)

Получает значение параметра "Value" (значение) элемента управления CSpinEdit.

```
int  Value()  const
```

Возвращаемое значение

Значение параметра "Value".

Value (метод Set)

Устанавливает значение параметра "Value" (значение) элемента управления CSpinEdit.

```
void  Value(
   const int  value      // значение
   )
```

Параметры

value

[in] Новое значение параметра "Value".

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

CreateEdit

Создает подчиненный элемент управления CEdit ("поле ввода") для элемента управления CSpinEdit.

```
virtual bool  CreateEdit()
```

Возвращаемое значение

true - в случае удачи, иначе - false

© 2000-2025, MetaQuotes Ltd.

CreateInc

Создает кнопку для увеличения значения (инкремент) элемента управления CSpinEdit.

```
virtual bool  CreateInc()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateDec

Создает кнопку для уменьшения значения (декремент) элемента управления CSpinEdit.

```
virtual bool  CreateDec()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickInc

Виртуальный обработчик внутреннего события "ClickInc" (клик мыши на кнопке увеличения значения) элемента управления CSpinEdit.

```
virtual bool  OnClickInc()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickDec

Виртуальный обработчик внутреннего события "ClickDec" (клик мыши на кнопке уменьшения значения) элемента управления CSpinEdit.

```
virtual bool  OnClickDec()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnChangeValue

Виртуальный обработчик внутреннего события "ChangeValue" (изменение текущего значения) элемента управления CSpinEdit.

```
virtual bool  OnChangeValue()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Класс CDialog

Класс CDialog является классом комбинированного элемента управления "Диалог".

Описание

Класс CDialog предназначен для визуального объединения группы функционально связанных разнородных элементов.

Декларация

```
class CDialog : public CWndContainer
```

Заголовок

```
#include <Controls\Dialog.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CDialog

Прямые потомки

CAppDialog

Методы класса по группам

Создание	
Create	Создает элемент управления
Обработка событий графика	
OnEvent	Обрабатывает все события графика
Настройка	
Caption	Получает/устанавливает значение свойства "Caption" (заголовок) элемента управления
Наполнение	
Add	Добавляет элемент управления в клиентскую область по указателю/ссылке
Подчиненные элементы управления	
CreateWhiteBorder	Создает подчиненный элемент (белая рамка) элемента управления

© 2000-2025, MetaQuotes Ltd.

Создание	
CreateBackground	Создает подчиненный элемент (подложка) элемента управления
CreateCaption	Создает подчиненный элемент (заголовок окна) элемента управления
CreateButtonClose	Создает подчиненный элемент (кнопка закрытия окна) элемента управления
CreateClientArea	Создает подчиненный элемент (клиентская область) элемента управления
Обработка событий подчиненных элементов управления	
OnClickCaption	Виртуальный обработчик внутреннего события "ClickCaption" элемента управления
OnClickButtonClose	Виртуальный обработчик внутреннего события "ClickButtonClose" элемента управления
Доступ к свойствам клиентской области	
ClientAreaVisible	Устанавливает флаг видимости подчиненного элемента (клиентская область) элемента управления
ClientAreaLeft	Получает координату X левой верхней точки клиентской области элемента управления
ClientAreaTop	Получает координату Y левой верхней точки клиентской области элемента управления
ClientAreaRight	Получает координату X правой нижней точки клиентской области элемента управления
ClientAreaBottom	Получает координату Y правой нижней точки клиентской области элемента управления
ClientAreaWidth	Получает ширину клиентской области элемента управления
ClientAreaHeight	Получает высоту клиентской области элемента управления
Обработка перетаскивания	
OnDialogDragStart	Виртуальный обработчик события "DialogDragStart" элемента управления
OnDialogDragProcess	Виртуальный обработчик события "DialogDragProcess" элемента управления
OnDialogDragEnd	Виртуальный обработчик события "DialogDragEnd" элемента управления

© 2000-2025, MetaQuotes Ltd.

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CDialog.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Caption (метод Get)

Получает значение свойства "Caption" (заголовок) элемента управления CDialog.

```
string  MinValue()  const
```

Возвращаемое значение

Значение свойства "Caption".

Caption (метод Set)

Устанавливает значение свойства "Caption" (заголовок) элемента управления CDialog.

```
bool  Caption(
   const string  text      // текст
   )
```

Параметры

text

[in] Новое значение свойства "Caption".

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Add

Добавляет элемент управления в клиентскую область по указателю.

```
bool  Add(
   CWnd   *control,         // указатель
   )
```

Параметры

control

[in] Указатель на элемент управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

Add

Добавляет элемент управления в клиентскую область по ссылке.

```
bool  Add(
   CWnd   &control,         // ссылка
   )
```

Параметры

control

[in] Ссылка на элемент управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateWhiteBorder

Создает подчиненный элемент (белая рамка) элемента управления CDialog.

```
virtual bool  CreateWhiteBorder()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateBackground

Создает подчиненный элемент (подложка) элемента управления CDialog.

```
virtual bool  CreateBackground()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateCaption

Создает подчиненный элемент (заголовок окна) элемента управления CDialog.

```
virtual bool  CreateCaption()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateButtonClose

Создает подчиненный элемент (кнопка закрытия окна) элемента управления CDialog.

```
virtual bool  CreateButtonClose()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateClientArea

Создает подчиненный элемент (клиентская область) элемента управления CDialog.

```
virtual bool  CreateClientArea()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickCaption

Виртуальный обработчик внутреннего события "ClickCaption" (клик мыши на заголовке окна) элемента управления CDialog.

```
virtual bool  OnClickCaption()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

OnClickButtonClose

Виртуальный обработчик внутреннего события "ClickButtonClose" (клик мыши на кнопке закрытия окна) элемента управления CDialog.

```
virtual bool  OnClickButtonClose()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ClientAreaVisible

Устанавливает флаг видимости подчиненного элемента (клиентская область) элемента управления CDialog.

```
bool  ClientAreaVisible(
   const bool  visible      // флаг видимости
   )
```

Параметры

visible

[in] Флаг видимости.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ClientAreaLeft

Получает координату X левой верхней точки клиентской области элемента управления CDialog.

```
int  ClientAreaLeft()
```

Возвращаемое значение

Координата X левой верхней точки клиентской области.

© 2000-2025, MetaQuotes Ltd.

ClientAreaTop

Получает координату Y левой верхней точки клиентской области элемента управления CDialog.

```
int  ClientAreaTop()
```

Возвращаемое значение

Координата Y левой верхней точки клиентской области.

© 2000-2025, MetaQuotes Ltd.

ClientAreaRight

Получает координату X правой нижней точки клиентской области элемента управления CDialog.

```
int  ClientAreaTop()
```

Возвращаемое значение

Координата X правой нижней точки клиентской области.

© 2000-2025, MetaQuotes Ltd.

ClientAreaBottom

Получает координату Y правой нижней точки клиентской области элемента управления CDialog.

```
int  ClientAreaBottom()
```

Возвращаемое значение

Координата Y правой нижней точки клиентской области.

© 2000-2025, MetaQuotes Ltd.

ClientAreaWidth

Получает ширину клиентской области элемента управления CDialog.

```
int  ClientAreaWidth()
```

Возвращаемое значение

Ширина клиентской области.

© 2000-2025, MetaQuotes Ltd.

ClientAreaHeight

Получает высоту клиентской области элемента управления CDialog.

```
int  ClientAreaHeight()
```

Возвращаемое значение

Высота клиентской области.

© 2000-2025, MetaQuotes Ltd.

OnDialogDragStart

Виртуальный обработчик события "DialogDragStart" (начало операции перетаскивания) элемента управления CDialog.

```
virtual bool  OnDialogDragStart()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "DialogDragStart" происходит при начале операции перетаскивания элемента управления CDialog.

© 2000-2025, MetaQuotes Ltd.

OnDialogDragProcess

Виртуальный обработчик события "DialogDragProcess" (операция перетаскивания) элемента управления CDialog.

```
virtual bool  OnDialogDragProcess()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "DialogDragProcess" происходит при перемещении элемента управления CDialog.

© 2000-2025, MetaQuotes Ltd.

OnDialogDragEnd

Виртуальный обработчик события "DialogDragEnd" (завершение операции перетаскивания) элемента управления CDialog.

```
virtual bool  OnDialogDragEnd()
```

Возвращаемое значение

true - если событие обработано, иначе - false.

Примечание

Событие "DialogDragEnd" происходит по завершении операции перетаскивания элемента управления CDialog.

© 2000-2025, MetaQuotes Ltd.

Класс CAppDialog

Класс CAppDialog является классом комбинированного элемента управления "Диалог приложения".

Описание

Класс CAppDialog предназначен для визуального объединения группы функционально связанных разнородных элементов в пределах одной MQL5-программы.

Декларация

```
class CAppDialog : public CDialog
```

Заголовок

```
#include <Controls\Dialog.mqh>
```

Иерархия наследования

CObject
  CWnd
    CWndContainer
      CDialog
        CAppDialog

Методы класса по группам

Создание и уничтожение	
Create	Создает элемент управления
Destroy	Уничтожает элемент управления
Обработка событий	
OnEvent	Обрабатывает все события
Запуск	
Run	Запускает элемент управления
Обработка событий графика	
ChartEvent	Обрабатывает все события графика
Настройка	
Minimized	Устанавливает флаг минимизации элемента управления
Сохранение/восстановление состояния	
IniFileSave	Сохраняет состояние элемента управления в файл

© 2000-2025, MetaQuotes Ltd.

Создание и уничтожение	
IniFileLoad	Загружает состояние элемента управления из файла
IniFileName	Устанавливает имя файла для сохранения/загрузки состояния элемента управления
IniFileExt	Устанавливает расширение файла для сохранения/загрузки состояния элемента управления
Инициализация	
CreateCommon	Метод инициализации общих настроек элемента управления
CreateExpert	Метод инициализации настроек элемента управления для работы в эксперте
CreateIndicator	Метод инициализации настроек элемента управления для работы в индикаторе
Подчиненные элементы управления	
CreateButtonMinMax	Создает вспомогательные элементы (кнопки минимизации/восстановления окна) элемента управления
Обработка событий подчиненных элементов управления	
OnClickButtonClose	Виртуальный обработчик внутреннего события "ClickButtonClose" элемента управления
OnClickButtonMinMax	Виртуальный обработчик внутреннего события "ClickButtonMinMax" элемента управления
Обработка внешних событий	
OnAnotherApplicationClose	Виртуальный обработчик внешних событий элемента управления
Методы	
Rebound	Устанавливает новые параметры прямоугольной области элемента управления из координат класса CRect
Minimize	Устанавливает окно элемента управления в минимизированное состояние
Maximize	Устанавливает окно элемента управления в развернутое состояние
CreateInstanceId	Создает уникальный префикс для имен объектов элемента управления
ProgramName	Получает имя программы, в которой используется элемент управления

© 2000-2025, MetaQuotes Ltd.

Создание и уничтожение	
SubwinOff	Получает смещение по оси Y подокна, в котором расположен элемент управления

Методы унаследованные от CObject

Prev, Prev, Next, Next, Type, Compare

Методы унаследованные от 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

Методы унаследованные от CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Методы унаследованные от CDialog

Caption, Caption, Add, Add

© 2000-2025, MetaQuotes Ltd.

Create

Создает элемент управления CAppDialog.

```
virtual bool  Create(
   const long     chart,       // идентификатор графика
   const string   name,        // имя
   const int      subwin,      // подокно графика
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Destroy

Метод деинициализации элемента управления CAppDialog.

```
virtual void  Destroy(
   const int  reason=REASON_PROGRAM    // код причины
)
```

Параметры

reason

[in] Код причины деинициализации. По умолчанию указывается REASON_PROGRAM.

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnEvent

Обрабатывает все события графика.

```
virtual bool  OnEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Run

Метод запуска элемента управления CAppDialog.

```
bool  Run()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

ChartEvent

Виртуальный обработчик событий элемента управления CAppDialog.

```
virtual bool  ChartEvent(
   const int      id,          // идентификатор
   const long&    lparam,      // параметр
   const double&  dparam,      // параметр
   const string&  sparam       // параметр
   )
```

Параметры

id

[in] Идентификатор события.

lparam

[in] Ссылка на параметр события типа long.

dparam

[in] Ссылка на параметр события типа double.

sparam

[in] Ссылка на параметр события типа string.

Возвращаемое значение

true - если событие обработано, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Minimized

Устанавливает значение флага "Minimized" (состояние окна) элемента управления CAppDialog.

```
bool  Minimized(
   const bool  flag      // состояние
   )
```

Параметры

flag

[in] Новое состояние.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

IniFileSave

Сохраняет состояние элемента управления CAppDialog в файл.

```
void  IniFileSave()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

IniFileLoad

Загружает состояние элемента управления CAppDialog в файл.

```
void  IniFileLoad()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

IniFileName

Получает имя файла для сохранения/загрузки состояния элемента управления CAppDialog.

```
virtual string  IniFileName()  const
```

Возвращаемое значение

Имя файла для сохранения/загрузки состояния элемента управления CAppDialog.

Примечание

Имя файла включает в себя имя индикатора/эксперта а также символ, на котором работает программа.

© 2000-2025, MetaQuotes Ltd.

IniFileExt

Получает расширение файла для сохранения/загрузки состояния элемента управления CAppDialog.

```
virtual string  IniFileExt()  const
```

Возвращаемое значение

Расширение файла для сохранения/загрузки состояния элемента управления CAppDialog.

© 2000-2025, MetaQuotes Ltd.

CreateCommon

Метод инициализации общих настроек элемента управления CAppDialog.

```
bool  CreateCommon(
   const long    chart,       // идентификатор графика
   const string  name,        // имя
   const int     subwin,      // подокно графика
   )
```

Параметры

chart

[in] Идентификатор графика, на котором создается элемент управления.

name

[in] Уникальное имя элемента управления.

subwin

[in] Подокно графика, в котором создается элемент управления.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateExpert

Метод инициализации настроек элемента управления CAppDialog для работы в эксперте.

```
bool  CreateExpert(
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateIndicator

Метод инициализации настроек элемента управления CAppDialog для работы в индикаторе.

```
bool  CreateIndicator(
   const int      x1,          // координата
   const int      y1,          // координата
   const int      x2,          // координата
   const int      y2           // координата
   )
```

Параметры

x1

[in] Значение координаты X левой верхней точки.

y1

[in] Значение координаты Y левой верхней точки.

x2

[in] Значение координаты X правой нижней точки.

y2

[in] Значение координаты Y правой нижней точки.

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateButtonMinMax

Создает вспомогательные элементы (кнопки минимизации/восстановления) элемента управления CAppDialog.

```
virtual void  CreateButtonMinMax()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnClickButtonClose

Виртуальный обработчик внутреннего события "ClickButtonClose" (клик мыши на кнопке закрытия) элемента управления CAppDialog.

```
virtual void  OnClickButtonClose()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnClickButtonMinMax

Виртуальный обработчик внутреннего события "ClickButtonMinMax" (клик мыши на кнопке минимизации/восстановления) элемента управления CAppDialog.

```
virtual void  OnClickButtonClose()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

OnAnotherApplicationClose

Виртуальный обработчик внешних событий элемента управления CAppDialog.

```
virtual void  OnAnotherApplicationClose()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Rebound

Устанавливает новые параметры прямоугольной области элемента управления CAppDialog из координат класса CRect.

```
bool  Rebound(
   const & CRect  rect      // класс прямоугольной области
   )
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Minimize

Устанавливает окно элемента управления CAppDialog в минимизированное (свернутое) состояние.

```
virtual void  Minimize()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

Maximize

Устанавливает окно элемента управления CAppDialog в развернутое состояние.

```
virtual void  Maximize()
```

Возвращаемое значение

true - в случае удачи, иначе - false.

© 2000-2025, MetaQuotes Ltd.

CreateInstanceId

Создает уникальный префикс для имен объектов элемента управления CAppDialog.

```
string  CreateInstanceId()
```

Возвращаемое значение

Префикс для имен объектов.

© 2000-2025, MetaQuotes Ltd.

ProgramName

Получает имя программы, в которой используется элемент управления CAppDialog.

```
string  ProgramName()
```

Возвращаемое значение

Имя MQL5-программы.

© 2000-2025, MetaQuotes Ltd.

SubwinOff

Получает смещение по оси Y подокна, в котором расположен элемент управления CAppDialog.

```
void  SubwinOff()
```

Возвращаемое значение

Нет.

© 2000-2025, MetaQuotes Ltd.

Переход с MQL4

Язык MQL5 является развитием своего предшественника - языка MQL4, на котором написано огромное множество индикаторов, скриптов и экспертов. Несмотря на то, что новый язык программирования максимально совместим с языком предыдущего поколения, все же есть ряд отличий между этими языками. И при переносе программ эти отличия нужно знать.

В этом разделе собрана информация, которая призвана облегчить адаптацию своих кодов тем программистам, которые хорошо знают MQL4, к новому языку MQL5.

Прежде всего необходимо отметить:

- Функции start(), init() и deinit() отсутствуют;
- Количество индикаторных буферов не ограничено;
- Загрузка dll происходит сразу после загрузки эксперта (или любой другой mql5-программы);
- Укороченная проверка логических условий;
- При выходе за пределы массива текущее выполнение прекращается (критически - с выводом ошибки);
- Приоритет операций как в C++;
- Работает неявное приведение типов (даже из строки в число);
- Локальные переменные автоматически не инициализируются (кроме строк);
- Обычные локальные массивы уничтожаются автоматически.

Специальные функции init, start и deinit

В языке MQL4 было только три предопределенные функции, которые могли присутствовать в коде индикатора, советника или скрипта (не принимая во внимание включаемые файлы *.mqh и файлы библиотек). Этих функций нет в MQL5, но есть их аналоги. В таблице представлено примерное соответствие функций.

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

Функции OnInit и OnDeinit выполняют ту же роль, что и функции init и deinit в MQL4 - они предназначены для размещения кода, который должен выполняться при инициализации и деинициализации mql5-программ. Вы можете либо просто переименовать эти функции соответствующим образом, либо оставить их как есть, но добавить вызов этих функций в соответствующих местах.

Пример:

```
void OnInit()
  {
//--- вызовем функцию при инициализации
   init();
```

© 2000-2025, MetaQuotes Ltd.

```
  }
void OnDeinit(const int reason)
  {
//--- вызовем функцию при деинициализации
   deinit();
//---
  }
```

Функция start заменяется на OnStart только в скриптах, а для эксперта и индикатора необходимо ее переименовать соответственно в OnTick и OnCalculate. Именно в этих трех функциях необходимо размещать код, который должен выполняться во время работы mql5-программы:

mql5-программа	основная функция
скрипт	OnStart
индикатор	OnCalculate
эксперт	OnTick

Если основная функция отсутствует в коде индикатора или скрипта, или название этой функции отличается от требуемого, то вызов такой функции не производится. То есть, если в исходном коде скрипта не будет содержаться функции OnStart, то такой код будет скомпилирован как советник.

Если в коде индикатора отсутствует функция OnCalculate, то компиляция такого индикатора невозможна.

Предопределенные переменные

В MQL5 нет таких предопределенных переменных как Ask, Bid, Bars. Переменные Digits и Point немного изменились в написании, как показано в таблице.

MQL4	MQL5
Digits	_Digits
Point	_Point
	_LastError
	_Period
	_Symbol
	_StopFlag
	_UninitReason

Доступ к таймсериям

© 2000-2025, MetaQuotes Ltd.

В MQL5 нет предопределенных таймсерий Open[], High[], Low[], Close[], Volume[] и Time[]. Необходимую глубину таймсерии теперь можно задавать самостоятельно с помощью соответствующих функций для доступа к таймсериям.

Советники (эксперты)

Эксперты в MQL5 не обязательно должны иметь функцию-обработчик события поступления нового тика OnTick, как это было в MQL4 (функция start в MQL4 вызывается на выполнение при поступлении нового тика), потому что теперь эксперты в MQL5 могут содержать предопределенные функции-обработчики несколько типов событий:

- OnTick – поступление нового тика;
- OnTimer – событие таймера;
- OnTrade - торговое событие;
- OnChartEvent – события ввода от клавиатуры и мышки, события перемещения графического объекта, событие окончания редактирования текста в поле ввода объекта LabelEdit;
- OnBookEvent – событие изменения состояния стакана цен (Depth of Market).

Пользовательские индикаторы

В MQL4 количество индикаторных буферов ограничено и не может быть более 8. В MQL5 такого ограничения нет, но следует помнить, что каждый индикаторный буфер требует выделения определенного объема оперативной памяти под его размещение в терминале, поэтому злоупотреблять открывшейся новой возможностью все же не следует.

Кроме того, в MQL4 было всего 6 типов отрисовки пользовательского индикатора, в MQL5 теперь 18 стилей рисования. И хотя названия видов отрисовки не изменились, сама идеология графического отображения индикаторов изменилась существенно.

Отличается также и направление индексации в индикаторных буферах. В MQL5 все буфера по умолчанию ведут себя как обычные массивы, то есть элемент с индексом 0 является самым старым в истории, при увеличении индекса мы продвигаемся от самых старых данных к самым последним.

Единственная функция по работе с пользовательскими индикаторами, которая сохранилась из MQL4 – это SetIndexBuffer. Но и ее вызов изменился, теперь требуется указать тип данных, которые будут храниться в массиве, связываемом с индикаторным буфером.

Кроме того, изменились и расширились свойства пользовательских индикаторов, появились функции доступа к таймсериям, поэтому необходимо полностью переосмыслить весь алгоритм расчета.

Графические объекты

Количество графических объектов в MQL5 существенно расширилось. Кроме того, позиционирование графических объектов во времени стало возможным с точностью до секунды на графике любого периода – теперь не производится округление точек привязок графических объектов с точностью до времени открытия бара на текущем ценовом графике.

© 2000-2025, MetaQuotes Ltd.

Для объектов Arrow, Text и Label появилась возможность указывать способ привязки, а для объектов Label, Button, Chart, Bitmap Label и Edit есть возможность задавать угол графика, к которому привязан объект.

© 2000-2025, MetaQuotes Ltd.

Список функций языка MQL5

Все функции MQL5 в алфавитном порядке.

Функция	Действие	Раздел
AccountInfoDouble	Возвращает значение типа double соответствующего свойства счета	Информация о счете
AccountInfoInteger	Возвращает значение целочисленного типа (bool,int или long) соответствующего свойства счета	Информация о счете
AccountInfoString	Возвращает значение типа string соответствующего свойства счета	Информация о счете
acos	Возвращает значение арккосинуса x в радианах	Математические функции
Alert	Выводит сообщение в отдельном окне	Общие функции
ArrayBsearch	Возвращает индекс первого найденного элемента в первом измерении массива	Операции с массивами
ArrayCompare	Возвращает результат сравнения двух массивов простых типов или пользовательских структур, не имеющих сложных объектов	Операции с массивами
ArrayCopy	Копирует один массив в другой	Операции с массивами
ArrayFill	Заполняет числовой массив указанным значением	Операции с массивами
ArrayFree	Освобождает буфер любого динамического массива и устанавливает размер нулевого измерения в 0 (ноль)	Операции с массивами
ArrayGetAsSeries	Проверяет направление индексации массива	Операции с массивами
ArrayInitialize	Устанавливает все элементы числового массива в одну величину	Операции с массивами
ArrayIsDynamic	Проверяет, является ли массив динамическим	Операции с массивами

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
ArrayIsSeries	Проверяет, является ли массив таймсерией	Операции с массивами
ArrayMaximum	Ищет максимальный элемент в первом измерении многомерного числового массива	Операции с массивами
ArrayMinimum	Ищет минимальный элемент в первом измерении многомерного числового массива	Операции с массивами
ArrayRange	Возвращает число элементов в указанном измерении массива	Операции с массивами
ArrayResize	Устанавливает новый размер в первом измерении массива	Операции с массивами
ArraySetAsSeries	Устанавливает направление индексирования в массиве	Операции с массивами
ArraySize	Возвращает количество элементов в массиве	Операции с массивами
ArraySort	Сортирует многомерный числовой массив по возрастанию значений в первом измерении	Операции с массивами
ArrayPrint	Выводит в журнал массив простого типа или простой структуры	Операции с массивами
ArrayInsert	Вставляет в массив-приемник из массива-источника указанное число элементов	Операции с массивами
ArrayRemove	Удаляет из массива указанное число элементов начиная с указанного индекса	Операции с массивами
ArrayReverse	Разворачивает в массиве указанное число элементов начиная с указанного индекса	Операции с массивами
ArraySwap	Обменивает между собой содержимое двух динамических массивов одного типа	Операции с массивами
asin	Возвращает значение арксинуса x в радианах	Математические функции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
atan	Возвращает арктангенс x в радианах	Математические функции
Bars	Возвращает количество баров в истории по соответствующим символу и периоду	Доступ к таймсериям и индикаторам
BarsCalculated	Возвращает количество рассчитанных данных в индикаторном буфере или -1 в случае ошибки (данные еще не рассчитаны)	Доступ к таймсериям и индикаторам
CalendarCountryById	Получает описание страны по её идентификатору	Экономический календарь
CalendarEventById	Получает описание события по его идентификатору	Экономический календарь
CalendarValueById	Получает описание значения события по его идентификатору	Экономический календарь
CalendarCountries	Получает массив описаний стран, доступных в Календаре	Экономический календарь
CalendarEventByCountry	Получает массив описаний всех событий, доступных в Календаре, по указанному коду страны	Экономический календарь
CalendarEventByCurrency	Получает массив описаний всех событий, доступных в Календаре, по указанной валюте	Экономический календарь
CalendarValueHistoryByEvent	Получает массив значений по всем событиям на заданном диапазоне времени по идентификатору события	Экономический календарь
CalendarValueHistory	Получает массив значений по всем событиям на заданном диапазоне времени с фильтром по стране и/или валюте	Экономический календарь
CalendarValueLastByEvent	Получает массив значений события по его ID с момента состояния базы Календаря с заданным change_id	Экономический календарь
CalendarValueLast	Получает массив значений по всем событиям с фильтрацией	Экономический календарь

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	по стране и/или валюте с момента состояния базы Календаря с заданным change_id	
ceil	Возвращает ближайшее сверху целое числовое значение	Математические функции
CharArrayToString	Преобразует код символа (ansi) в односимвольную строку	Преобразование данных
ChartApplyTemplate	Применяет к указанному графику шаблон из указанного файла	Операции с графиками
ChartClose	Закрывает указанный график	Операции с графиками
ChartFirst	Возвращает идентификатор первого графика клиентского терминала	Операции с графиками
ChartGetDouble	Возвращает значение соответствующего свойства указанного графика	Операции с графиками
ChartGetInteger	Возвращает целочисленное значение соответствующего свойства указанного графика	Операции с графиками
ChartGetString	Возвращает строковое значение соответствующего свойства указанного графика	Операции с графиками
ChartID	Возвращает идентификатор текущего графика	Операции с графиками
ChartIndicatorAdd	Добавляет на указанное окно графика индикатор с указанным хэндлом.	Операции с графиками
ChartIndicatorDelete	Удаляет с указанного окна графика индикатор с указанным именем	Операции с графиками
ChartIndicatorGet	Возвращает хэндл индикатора с указанным коротким именем на указанном окне графика	Операции с графиками
ChartIndicatorName	Возвращает короткое имя индикатора по номеру в списке индикаторов на указанном окне графика.	Операции с графиками

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
ChartIndicatorsTotal	Возвращает количество всех индикаторов, присоединенных к указанному окну графика.	Операции с графиками
ChartNavigate	Осуществляет сдвиг указанного графика на указанное количество баров относительно указанной позиции графика	Операции с графиками
ChartNext	Возвращает идентификатор графика, следующего за указанным	Операции с графиками
ChartOpen	Открывает новый график с указанным символом и периодом	Операции с графиками
CharToString	Преобразование кода символа в односимвольную строку	Преобразование данных
ChartPeriod	Возвращает значение периода указанного графика	Операции с графиками
ChartPriceOnDropped	Возвращает ценовую координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт	Операции с графиками
ChartRedraw	Вызывает принудительную перерисовку указанного графика	Операции с графиками
ChartSaveTemplate	Сохраняет текущие настройки графика в шаблон с указанным именем	Операции с графиками
ChartScreenShot	Делает снимок указанного графика в формате GIF, PNG или BMP в зависимости от указанного расширения	Операции с графиками
ChartSetDouble	Задает значение типа double соответствующего свойства указанного графика	Операции с графиками
ChartSetInteger	Задает значение целочисленного типа (datetime, int, color, bool или char) соответствующего свойства указанного графика	Операции с графиками

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
ChartSetString	Задает значение типа string соответствующего свойства указанного графика	Операции с графиками
ChartSetSymbolPeriod	Меняет значения символа и периода указанного графика	Операции с графиками
ChartSymbol	Возвращает имя символа указанного графика	Операции с графиками
ChartTimeOnDropped	Возвращает временную координату, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт	Операции с графиками
ChartTimePriceToXY	Преобразует координаты графика из представления время/цена в координаты по оси X и Y	Операции с графиками
ChartWindowFind	Возвращает номер подокна, в котором находится индикатор	Операции с графиками
ChartWindowOnDropped	Возвращает номер подокна графика, на которое брошен мышкой данный эксперт, скрипт, объект или индикатор	Операции с графиками
ChartXOnDropped	Возвращает координату по оси X, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт	Операции с графиками
ChartXYToTimePrice	Преобразует координаты X и Y графика в значения время и цена	Операции с графиками
ChartYOnDropped	Возвращает координату по оси Y, соответствующую точке, в которой брошен мышкой данный эксперт или скрипт	Операции с графиками
CheckPointer	Возвращает тип указателя объекта	Общие функции
CLBufferCreate	Создает буфер OpenCL	Работа с OpenCL
CLBufferFree	Удаляет буфер OpenCL	Работа с OpenCL
CLBufferRead	Читает буфер OpenCL в массив и возвращает количество прочитанных элементов	Работа с OpenCL

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
CLBufferWrite	Записывает массив в буфер OpenCL и возвращает количество записанных элементов	Работа с OpenCL
CLContextCreate	Создает контекст OpenCL	Работа с OpenCL
CLContextFree	Удаляет контекст OpenCL	Работа с OpenCL
CLExecute	Выполняет OpenCL программу	Работа с OpenCL
CLGetDeviceInfo	Получает свойство устройства из OpenCL драйвера	Работа с OpenCL
CLGetInfoInteger	Возвращает значение целочисленного свойства для OpenCL-объекта или устройства	Работа с OpenCL
CLHandleType	Возвращает тип OpenCL хендла в виде значения из перечисления ENUM_OPENCL_HANDLE_TYPE	Работа с OpenCL
CLKernelCreate	Создает функцию запуска OpenCL	Работа с OpenCL
CLKernelFree	Удаляет функцию запуска OpenCL	Работа с OpenCL
CLProgramCreate	Создает OpenCL программу из исходного кода	Работа с OpenCL
CLProgramFree	Удаляет OpenCL программу	Работа с OpenCL
CLSetKernelArg	Выставляет параметр для функции OpenCL	Работа с OpenCL
CLSetKernelArgMem	Выставляет буфер OpenCL в качестве параметра функции OpenCL	Работа с OpenCL
ColorToARGB	Преобразует тип color в тип uint для получения ARGB-представления цвета.	Преобразование данных
ColorToString	Преобразует значение цвета в строку вида "R,G,B"	Преобразование данных
Comment	Выводит сообщение в левый верхний угол ценового графика	Общие функции
CopyBuffer	Получает в массив данные указанного буфера от	Доступ к таймсериям и индикаторам

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	указанного индикатора	
CopyClose	Получает в массив исторические данные по цене закрытия баров по соответствующим символу и периоду	Доступ к таймсериям и индикаторам
CopyHigh	Получает в массив исторические данные по максимальной цене баров по соответствующим символу и периоду	Доступ к таймсериям и индикаторам
CopyLow	Получает в массив исторические данные по минимальной цене баров по соответствующим символу и периоду	Доступ к таймсериям и индикаторам
CopyOpen	Получает в массив исторические данные по цене открытия баров по соответствующим символу и периоду	Доступ к таймсериям и индикаторам
CopyRates	Получает в массив исторические данные структуры Rates для указанных символа и периода	Доступ к таймсериям и индикаторам
CopyRealVolume	Получает в массив исторические данные по торговым объемам для соответствующих символа и периода	Доступ к таймсериям и индикаторам
CopySpread	Получает в массив исторические данные по спредам для соответствующих символа и периода	Доступ к таймсериям и индикаторам
CopyTicks	Получает в массив тики, накопленные терминалом за текущую рабочую сессию	Доступ к таймсериям и индикаторам
CopyTickVolume	Получает в массив исторические данные по тиковым объемам для соответствующих символа и периода	Доступ к таймсериям и индикаторам

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
CopyTime	Получает в массив исторические данные по времени открытия баров по соответствующим символу и периоду	Доступ к таймсериям и индикаторам
cos	Возвращает косинус числа	Математические функции
CryptDecode	Производит обратное преобразование данных массива	Общие функции
CryptEncode	Преобразует данные массива-источника в массив-приемник указанным методом	Общие функции
CustomSymbolCreate	Создает пользовательский символ с указанным именем в указанной группе	Пользовательские символы
CustomSymbolDelete	Удаляет пользовательский символ с указанным именем	Пользовательские символы
CustomSymbolSetInteger	Устанавливает для пользовательского символа значение свойства целочисленного типа	Пользовательские символы
CustomSymbolSetDouble	Устанавливает для пользовательского символа значение свойства вещественного типа	Пользовательские символы
CustomSymbolSetString	Устанавливает для пользовательского символа значение свойства строкового типа	Пользовательские символы
CustomSymbolSetMarginRate	Устанавливает для пользовательского символа коэффициенты взимания маржи в зависимости от типа и направления ордера	Пользовательские символы
CustomSymbolSetSessionQuote	Устанавливает время начала и время окончания указанной котировочной сессии для указанных символа и дня недели	Пользовательские символы
CustomSymbolSetSessionTrade	Устанавливает время начала и время окончания указанной	Пользовательские символы

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	торговой сессии для указанных символа и дня недели	
CustomRatesDelete	Удаляет все бары в указанном временном интервале из ценовой истории пользовательского инструмента	Пользовательские символы
CustomRatesReplace	Полностью заменяет ценовую историю пользовательского инструмента в указанном временном интервале данными из массива типа MqlRates	Пользовательские символы
CustomRatesUpdate	Добавляет в историю пользовательского инструмента отсутствующие бары и заменяет существующие бары данными из массива типа MqlRates	Пользовательские символы
CustomTicksAdd	Добавляет в ценовую историю пользовательского инструмента данные из массива типа MqlTick. Пользовательский символ должен быть выбран в окне MarketWatch (Обзор рынка)	Пользовательские символы
CustomTicksDelete	Удаляет все тики в указанном временном интервале из ценовой истории пользовательского инструмента	Пользовательские символы
CustomTicksReplace	Полностью заменяет ценовую историю пользовательского инструмента в указанном временном интервале данными из массива типа MqlTick	Пользовательские символы
CustomBookAdd	Передает состояние стакана цен по пользовательскому инструменту	Пользовательские символы
DatabaseOpen	Открывает или создаёт базу данных в указанном файле	Работа с базами данных
DatabaseClose	Закрывает базу данных	Работа с базами данных

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
DatabaseImport	Импортирует в таблицу данные из файла	Работа с базами данных
DatabaseExport	Экспортирует таблицу или результат выполнения SQL-запроса в CSV-файл	Работа с базами данных
DatabasePrint	Печатает таблицу или результат выполнения SQL-запроса в журнал экспертов	Работа с базами данных
DatabaseTableExists	Проверяет наличие таблицы в базе данных	Работа с базами данных
DatabaseExecute	Исполняет запрос к указанной базе данных	Работа с базами данных
DatabasePrepare	Создает хендл запроса, который затем может быть исполнен с помощью DatabaseRead()	Работа с базами данных
DatabaseReset	Сбрасывает запрос в начальное состояние, как после вызова DatabasePrepare()	Работа с базами данных
DatabaseBind	Устанавливает значение параметра в запросе	Работа с базами данных
DatabaseBindArray	Устанавливает массив в качестве значения параметра	Работа с базами данных
DatabaseRead	Выполняет переход к следующей записи в результате запроса	Работа с базами данных
DatabaseReadBind	Выполняет переход к следующей записи и читает из неё данные в структуру	Работа с базами данных
DatabaseFinalize	Удаляет запрос, созданный в DatabasePrepare()	Работа с базами данных
DatabaseTransactionBegin	Начинает выполнение транзакции	Работа с базами данных
DatabaseTransactionCommit	Завершает выполнение транзакции	Работа с базами данных
DatabaseTransactionRollback	Выполняет откат транзакций	Работа с базами данных
DatabaseColumnsCount	Получает количество полей в запросе	Работа с базами данных

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
DatabaseColumnName	Получает имя поля по номеру	Работа с базами данных
DatabaseColumnType	Получает тип поля по номеру	Работа с базами данных
DatabaseColumnSize	Получает размер поля в байтах	Работа с базами данных
DatabaseColumnText	Получает из текущей записи значение поля в виде строки	Работа с базами данных
DatabaseColumnInteger	Получает из текущей записи значение типа int	Работа с базами данных
DatabaseColumnLong	Получает из текущей записи значение типа long	Работа с базами данных
DatabaseColumnDouble	Получает из текущей записи значение типа double	Работа с базами данных
DatabaseColumnBlob	Получает из текущей записи значение поля в виде массива	Работа с базами данных
DebugBreak	Программная точка останова при отладке	Общие функции
Digits	Возвращает количество десятичных знаков после запятой, определяющее точность измерения цены символа текущего графика	Проверка состояния
DoubleToString	Преобразование числового значения в текстовую строку с указанной точностью	Преобразование данных
DXContextCreate	Создает графический контекст для отрисовки кадров указанного размера	Работа с DirectX
DXContextSetSize	Изменяет размер кадра графического контекста, созданного в DXContextCreate()	Работа с DirectX
DXContextSetSize	Получает размер кадра графического контекста, созданного в DXContextCreate()	Работа с DirectX
DXContextClearColors	Устанавливает для буфера отрисовки все пиксели в указанный цвет	Работа с DirectX
DXContextClearDepth	Очищает буфер глубины	Работа с DirectX

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
DXContextGetColors	Получает из графического контекста изображение указанного размера и смещения	Работа с DirectX
DXContextGetDepth	Получает буфер глубины отрисованного кадра	Работа с DirectX
DXBufferCreate	Создает буфер указанного типа на основе массива данных	Работа с DirectX
DXTextureCreate	Создает 2-мерную текстуру из прямоугольника заданного размера, вырезанного из переданного изображения	Работа с DirectX
DXInputCreate	Создает входные параметры шейдера	Работа с DirectX
DXInputSet	Устанавливает входные параметры шейдера	Работа с DirectX
DXShaderCreate	Создает шейдер указанного типа	Работа с DirectX
DXShaderSetLayout	Устанавливает разметку вершин для вершинного шейдера	Работа с DirectX
DXShaderInputsSet	Устанавливает входные параметры шейдера	Работа с DirectX
DXShaderTexturesSet	Устанавливает текстуры для шейдера	Работа с DirectX
DXDraw	Отрисовывает вершины вершинного буфера, установленного в DXBufferSet()	Работа с DirectX
DXDrawIndexed	Отрисовывает графические примитивы, описанные индексным буфером из DXBufferSet()	Работа с DirectX
DXPrimiveTopologySet	Устанавливает тип примитивов для отрисовки с помощью DXDrawIndexed()	Работа с DirectX
DXBufferSet	Устанавливает буфер для текущей отрисовки	Работа с DirectX
DXShaderSet	Устанавливает шейдер для отрисовки	Работа с DirectX

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
DXHandleType	Возвращает тип хендла	Работа с DirectX
DXRelease	Освобождает хендл	Работа с DirectX
EnumToString	Преобразование значения перечисления любого типа в строку	Преобразование данных
EventChartCustom	Генерирует пользовательское событие для указанного графика	Работа с событиями
EventKillTimer	Останавливает на текущем графике генерацию событий по таймеру	Работа с событиями
EventSetMillisecondTimer	Запускает генератор событий таймера высокого разрешения с периодом менее 1 секунды для текущего графика	Работа с событиями
EventSetTimer	Запускает генератор событий таймера с указанной периодичностью для текущего графика	Работа с событиями
exp	Возвращает экспоненту числа	Математические функции
ExpertRemove	Прекращает работу эксперта и выгружает его с графика	Общие функции
fabs	Возвращает абсолютное значение (значение по модулю) переданного ей числа	Математические функции
FileClose	Закрывает ранее открытый файл	Файловые операции
FileCopy	Копирует исходный файл из локальной или общей папки в другой файл	Файловые операции
FileDelete	Удаляет указанный файл	Файловые операции
FileFindClose	Закрывает хэндл поиска	Файловые операции
FileFindFirst	Начинает перебор файлов в соответствующей директории в соответствии с указанным фильтром	Файловые операции
FileFindNext	Продолжает поиск, начатый функцией FileFindFirst()	Файловые операции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
FileFlush	Сброс на диск всех данных, оставшихся в файловом буфере ввода-вывода	Файловые операции
FileGetInteger	Получает целочисленное свойство файла	Файловые операции
FileIsEnding	Определяет конец файла в процессе чтения	Файловые операции
FileIsExist	Проверяет существование файла	Файловые операции
FileIsLineEnding	Определяет конец строки в текстовом файле в процессе чтения	Файловые операции
FileMove	Перемещает или переименовывает файл	Файловые операции
FileOpen	Открывает файл с указанным именем и указанными флагам	Файловые операции
FileReadArray	Читает массивы любых типов, кроме строковых (может быть массив структур, не содержащих строки и динамические массивы), из бинарного файла с текущего положения файлового указателя	Файловые операции
FileReadBool	Читает из файла типа CSV строку от текущего положения до разделителя (либо до конца текстовой строки) и преобразует прочитанную строку в значение типа bool	Файловые операции
FileReadDatetime	Читает из файла типа CSV строку одного из форматов: "YYYY.MM.DD HH:MM:SS", "YYYY.MM.DD" или "HH:MM:SS" - и преобразует ее в значение типа datetime	Файловые операции
FileReadDouble	Читает число двойной точности с плавающей точкой (double) из бинарного файла с текущего положения файлового указателя	Файловые операции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
FileReadFloat	Читает из текущего положения файлового указателя значение типа float	Файловые операции
FileReadInteger	Читает из бинарного файла значение типа int, short или char в зависимости от указанной длины в байтах	Файловые операции
FileReadLong	Читает из текущего положения файлового указателя значение типа long	Файловые операции
FileReadNumber	Читает из файла типа CSV строку от текущего положения до разделителя (либо до конца текстовой строки) и преобразует прочитанную строку в значение типа double	Файловые операции
FileReadString	Читает из файла строку с текущего положения файлового указателя	Файловые операции
FileReadStruct	Считывает из бинарного файла содержимое в структуру, переданную в качестве параметра	Файловые операции
FileSeek	Перемещает положение файлового указателя на указанное количество байт относительно указанного положения	Файловые операции
FileSize	Возвращает размер соответствующего открытого файла	Файловые операции
FileTell	Возвращает текущее положение файлового указателя соответствующего открытого файла	Файловые операции
FileWrite	Записывает данные в файл типа CSV или TXT	Файловые операции
FileWriteArray	Записывает в файл типа BIN массивы любых типов, кроме строковых	Файловые операции
FileWriteDouble	Записывает в двоичный файл значение параметра типа	Файловые операции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	double с текущего положения файлового указателя	
FileWriteFloat	Записывает в двоичный файл значение параметра типа float с текущего положения файлового указателя	Файловые операции
FileWriteInteger	Записывает в двоичный файл значение параметра типа int с текущего положения файлового указателя	Файловые операции
FileWriteLong	Записывает в двоичный файл значение параметра типа long с текущего положения файлового указателя	Файловые операции
FileWriteString	Записывает в файл типа BIN или TXT значение параметра типа string с текущего положения файлового указателя	Файловые операции
FileWriteStruct	Записывает в двоичный файл содержимое структуры, переданной в качестве параметра, с текущего положения файлового указателя	Файловые операции
floor	Возвращает ближайшее снизу целое числовое значение	Математические функции
fmax	Возвращает максимальное из двух числовых значений	Математические функции
fmin	Возвращает минимальное из двух числовых значений	Математические функции
fmod	Возвращает вещественный остаток от деления двух чисел	Математические функции
FolderClean	Удаляет все файлы в указанной папке	Файловые операции
FolderCreate	Создает директорию в папке Files (в зависимости от значения common_flag)	Файловые операции
FolderDelete	Удаляет указанную директорию. Если папка не	Файловые операции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	пуста, то она не может быть удалена	
FrameAdd	Добавляет фрейм с данными	Работа с результатами оптимизации
FrameFilter	Устанавливает фильтр чтения фреймов и переводит указатель на начало	Работа с результатами оптимизации
FrameFirst	Переводит указатель чтения фреймов в начало и сбрасывает ранее установленный фильтр	Работа с результатами оптимизации
FrameInputs	Получает input-параметры, на которых сформирован фрейм	Работа с результатами оптимизации
FrameNext	Читает фрейм и перемещает указатель на следующий	Работа с результатами оптимизации
GetLastError	Возвращает значение последней ошибки	Проверка состояния
GetPointer	Возвращает указатель объекта	Общие функции
GetTickCount	Возвращает количество миллисекунд, прошедших с момента старта системы	Общие функции
GlobalVariableCheck	Проверяет существование глобальной переменной с указанным именем	Глобальные переменные терминала
GlobalVariableDel	Удаляет глобальную переменную	Глобальные переменные терминала
GlobalVariableGet	Запрашивает значение глобальной переменной	Глобальные переменные терминала
GlobalVariableName	Возвращает имя глобальной переменной по порядковому номеру в списке глобальных переменных	Глобальные переменные терминала
GlobalVariablesDeleteAll	Удаляет глобальные переменные с указанным префиксом в имени	Глобальные переменные терминала
GlobalVariableSet	Устанавливает новое значение глобальной переменной	Глобальные переменные терминала
GlobalVariableSetOnCondition	Устанавливает новое значение существующей глобальной	Глобальные переменные терминала

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	переменной по условию	
GlobalVariablesFlush	Принудительно записывает содержимое всех глобальных переменных на диск	Глобальные переменные терминала
GlobalVariablesTotal	Возвращает общее количество глобальных переменных	Глобальные переменные терминала
GlobalVariableTemp	Устанавливает новое значение глобальной переменной, которая существует только на время текущего сеанса работы терминала	Глобальные переменные терминала
GlobalVariableTime	Возвращает время последнего доступа к глобальной переменной	Глобальные переменные терминала
HistoryDealGetDouble	Возвращает запрошенное свойство сделки в истории (double)	Торговые функции
HistoryDealGetInteger	Возвращает запрошенное свойство сделки в истории (datetime или int)	Торговые функции
HistoryDealGetString	Возвращает запрошенное свойство сделки в истории (string)	Торговые функции
HistoryDealGetTicket	Выбирает сделку для дальнейшей обработки и возвращает тикет сделки в истории	Торговые функции
HistoryDealSelect	Выбирает в истории сделку для дальнейших обращений к ней через соответствующие функции	Торговые функции
HistoryDealsTotal	Возвращает количество сделок в истории	Торговые функции
HistoryOrderGetDouble	Возвращает запрошенное свойство ордера в истории (double)	Торговые функции
HistoryOrderGetInteger	Возвращает запрошенное свойство ордера в истории (datetime или int)	Торговые функции
HistoryOrderGetString	Возвращает запрошенное свойство ордера в истории	Торговые функции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	(string)	
HistoryOrderGetTicket	Возвращает тикет соответствующего ордера в истории	Торговые функции
HistoryOrderSelect	Выбирает в истории ордер для дальнейшей работы с ним	Торговые функции
HistoryOrdersTotal	Возвращает количество ордеров в истории	Торговые функции
HistorySelect	Запрашивает историю сделок и ордеров за указанный период серверного времени	Торговые функции
HistorySelectByPosition	Запрашивает историю сделок и ордеров с указанным идентификатором позиции.	Торговые функции
iBars	Возвращает количество баров в истории по соответствующему символу и периоду	Доступ к таймсериям и индикаторам
iBarShift	Возвращает индекс бара, в который попадает указанное время	Доступ к таймсериям и индикаторам
iClose	Возвращает значение цены закрытия бара (указанного параметром shift) соответствующего графика	Доступ к таймсериям и индикаторам
iHigh	Возвращает значение максимальной цены бара (указанного параметром shift) соответствующего графика	Доступ к таймсериям и индикаторам
iHighest	Возвращает индекс наибольшего найденного значения (смещение относительно текущего бара) соответствующего графика	Доступ к таймсериям и индикаторам
iLow	Возвращает значение минимальной цены бара (указанного параметром shift) соответствующего графика	Доступ к таймсериям и индикаторам
iLowest	Возвращает индекс наименьшего найденного значения (смещение	Доступ к таймсериям и индикаторам

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	относительно текущего бара) соответствующего графика	
iOpen	Возвращает значение цены открытия бара (указанного параметром shift) соответствующего графика	Доступ к таймсериям и индикаторам
iTime	Возвращает значение времени открытия бара (указанного параметром shift) соответствующего графика	Доступ к таймсериям и индикаторам
iTickVolume	Возвращает значение тикового объема бара (указанного параметром shift) соответствующего графика	Доступ к таймсериям и индикаторам
iRealVolume	Возвращает значение реального объема бара (указанного параметром shift) соответствующего графика	Доступ к таймсериям и индикаторам
iVolume	Возвращает значение тикового объема бара (указанного параметром shift) соответствующего графика	Доступ к таймсериям и индикаторам
iSpread	Возвращает значение спреда бара (указанного параметром shift) соответствующего графика	Доступ к таймсериям и индикаторам
iAD	Accumulation/Distribution	Технические индикаторы
iADX	Average Directional Index	Технические индикаторы
iADXWilder	Average Directional Index by Welles Wilder	Технические индикаторы
iAlligator	Alligator	Технические индикаторы
iAMA	Adaptive Moving Average	Технические индикаторы
iAO	Awesome Oscillator	Технические индикаторы
iATR	Average True Range	Технические индикаторы
iBands	Bollinger Bands®	Технические индикаторы
iBearsPower	Bears Power	Технические индикаторы
iBullsPower	Bulls Power	Технические индикаторы

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
iBWMFI	Market Facilitation Index by Bill Williams	Технические индикаторы
iCCI	Commodity Channel Index	Технические индикаторы
iChaikin	Chaikin Oscillator	Технические индикаторы
iCustom	пользовательский индикатор	Технические индикаторы
iDEMA	Double Exponential Moving Average	Технические индикаторы
iDeMarker	DeMarker	Технические индикаторы
iEnvelopes	Envelopes	Технические индикаторы
iForce	Force Index	Технические индикаторы
iFractals	Fractals	Технические индикаторы
iFrAMA	Fractal Adaptive Moving Average	Технические индикаторы
iGator	Gator Oscillator	Технические индикаторы
iIchimoku	Ichimoku Kinko Hyo	Технические индикаторы
iMA	Moving Average	Технические индикаторы
iMACD	Moving Averages Convergence-Divergence	Технические индикаторы
iMFI	Money Flow Index	Технические индикаторы
iMomentum	Momentum	Технические индикаторы
IndicatorCreate	Возвращает хэндл указанного технического индикатора, созданного на основе массива параметров типа MqlParam	Доступ к таймсериям и индикаторам
IndicatorParameters	Возвращает по указанному хэндлу количество входных параметров индикатора, а также сами значения и тип параметров	Доступ к таймсериям и индикаторам
IndicatorRelease	Удаляет хэндл индикатора и освобождает расчетную часть индикатора, если ею больше никто не пользуется	Доступ к таймсериям и индикаторам
IndicatorSetDouble	Задает значение свойства индикатора, имеющего тип double	Пользовательские индикаторы

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
IndicatorSetInteger	Задает значение свойства индикатора,имеющего тип int	Пользовательские индикаторы
IndicatorSetString	Задает значение свойства индикатора, имеющего тип string	Пользовательские индикаторы
IntegerToString	Преобразование значения целого типа в строку указанной длины	Преобразование данных
iOBV	On Balance Volume	Технические индикаторы
iOsMA	Moving Average of Oscillator (MACD histogram)	Технические индикаторы
iRSI	Relative Strength Index	Технические индикаторы
iRVI	Relative Vigor Index	Технические индикаторы
iSAR	Parabolic Stop And Reverse System	Технические индикаторы
IsStopped	Возвращает true, если поступила команда завершить выполнение mql5-программы	Проверка состояния
iStdDev	Standard Deviation	Технические индикаторы
iStochastic	Stochastic Oscillator	Технические индикаторы
iTEMA	Triple Exponential Moving Average	Технические индикаторы
iTriX	Triple Exponential Moving Averages Oscillator	Технические индикаторы
iVIDyA	Variable Index Dynamic Average	Технические индикаторы
iVolumes	Volumes	Технические индикаторы
iWPR	Williams' Percent Range	Технические индикаторы
log	Возвращает натуральный логарифм	Математические функции
log10	Возвращает логарифм числа по основанию 10	Математические функции
MarketBookAdd	Обеспечивает открытие стакана цен по указанному инструменту, а также производит подписку на получение извещений об изменении указанного стакана	Получение рыночной информации

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
MarketBookGet	Возвращает массив структур типа MqlBookInfo, содержащий записи стакана цен указанного символа	Получение рыночной информации
MarketBookRelease	Обеспечивает закрытие стакана цен по указанному инструменту, а также отменяет подписку на получение извещений об изменении указанного стакана	Получение рыночной информации
MathAbs	Возвращает абсолютное значение (значение по модулю) переданного ей числа	Математические функции
MathArccos	Возвращает значение арккосинуса x в радианах	Математические функции
MathArcsin	Возвращает значение арксинуса x в радианах	Математические функции
MathArctan	Возвращает арктангенс x в радианах	Математические функции
MathCeil	Возвращает ближайшее сверху целое числовое значение	Математические функции
MathCos	Возвращает косинус числа	Математические функции
MathExp	Возвращает экспоненту числа	Математические функции
MathFloor	Возвращает ближайшее снизу целое числовое значение	Математические функции
MathIsValidNumber	Проверяет корректность действительного числа	Математические функции
MathLog	Возвращает натуральный логарифм	Математические функции
MathLog10	Возвращает логарифм числа по основанию 10	Математические функции
MathMax	Возвращает максимальное из двух числовых значений	Математические функции
MathMin	Возвращает минимальное из двух числовых значений	Математические функции
MathMod	Возвращает вещественный остаток от деления двух чисел	Математические функции
MathPow	Возводит основание в указанную степень	Математические функции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
MathRand	Возвращает псевдослучайное целое число в диапазоне от 0 до 32767	Математические функции
MathRound	Округляет число до ближайшего целого	Математические функции
MathSin	Возвращает синус числа	Математические функции
MathSqrt	Возвращает квадратный корень	Математические функции
MathSrand	Устанавливает начальное состояние генератора псевдослучайных целых чисел	Математические функции
MathTan	Возвращает тангенс числа	Математические функции
MessageBox	Создает и отображает окно сообщений, а также управляет им	Общие функции
MQLInfoInteger	Возвращает значение целого типа соответствующего свойства запущенной mql5-программы	Проверка состояния
MQLInfoString	Возвращает значение типа string соответствующего свойства запущенной mql5-программы	Проверка состояния
MT5Initialize	Устанавливает соединение с терминалом MetaTrader 5	MetaTrader для Python
MT5Shutdown	Закрывает ранее установленное подключение к терминалу MetaTrader 5	MetaTrader для Python
MT5TerminalInfo	Получает состояние и параметры подключенного терминала MetaTrader 5	MetaTrader для Python
MT5Version	Возвращает версию терминала MetaTrader 5	MetaTrader для Python
MT5CopyRatesFrom	Получает бары из терминала MetaTrader 5, начиная с указанной даты	MetaTrader для Python
MT5CopyRatesFromPos	Получает бары из терминала MetaTrader 5, начиная с указанного индекса	MetaTrader для Python

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
MT5CopyRatesRange	Получает бары в указанном диапазоне дат из терминала MetaTrader 5	MetaTrader для Python
MT5CopyTicksFrom	Получает тики из терминала MetaTrader 5, начиная с указанной даты	MetaTrader для Python
MT5CopyTicksRange	Получает тики за указанный диапазон дат из терминала MetaTrader 5	MetaTrader для Python
NormalizeDouble	Округление числа с плавающей точкой до указанной точности	Преобразование данных
ObjectCreate	Создает объект заданного типа на указанном графике	Графические объекты
ObjectDelete	Удаляет объект с указанным именем с указанного графика (с указанного подокна графика)	Графические объекты
ObjectFind	Ищет по имени объект с указанным идентификатором	Графические объекты
ObjectGetDouble	Возвращает значение типа double соответствующего свойства объекта	Графические объекты
ObjectGetInteger	Возвращает целочисленное значение соответствующего свойства объекта	Графические объекты
ObjectGetString	Возвращает значение типа string соответствующего свойства объекта	Графические объекты
ObjectGetTimeByValue	Возвращает значение времени для указанного значения цены объекта	Графические объекты
ObjectGetValueByTime	Возвращает ценовое значение объекта для указанного времени	Графические объекты
ObjectMove	Изменяет координаты указанной точки привязки объекта	Графические объекты
ObjectName	Возвращает имя объекта соответствующего типа в указанном графике (указанном подокне графика)	Графические объекты

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
ObjectsDeleteAll	Удаляет все объекты указанного типа с указанного графика (с указанного подокна графика)	Графические объекты
ObjectSetDouble	Устанавливает значение соответствующего свойства объекта	Графические объекты
ObjectSetInteger	Устанавливает значение соответствующего свойства объекта	Графические объекты
ObjectSetString	Устанавливает значение соответствующего свойства объекта	Графические объекты
ObjectsTotal	Возвращает количество объектов указанного типа в указанном графике (указанном подокне графика)	Графические объекты
OnStart	Вызывается в скрипте при наступлении события Start для выполнения действий, заложенных в скрипт	Обработка событий
OnInit	Вызывается в индикаторах и экспертах при наступлении события Init для инициализации запущенной MQL5-программы	Обработка событий
OnDeinit	Вызывается в индикаторах и экспертах при наступлении события Deinit для деинициализации запущенной MQL5-программы	Обработка событий
OnTick	Вызывается в экспертах при наступлении события NewTick для обработки новой котировки	Обработка событий
OnCalculate	Вызывается в индикаторах при наступлении события Calculate для обработки изменения ценовых данных	Обработка событий
OnTimer	Вызывается в индикаторах и экспертах при наступлении периодического события	Обработка событий

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	Timer, которое с заданным интервалом времени генерируется терминалом	
OnTrade	Вызывается в экспертах при наступлении события Trade, которое генерируется при завершении торговой операции на торговом сервере	Обработка событий
OnTradeTransaction	Вызывается в экспертах при наступлении события TradeTransaction для обработки результатов выполнения торгового запроса	Обработка событий
OnBookEvent	Вызывается в экспертах при наступлении события BookEvent для обработки изменений в стакане заявок	Обработка событий
OnChartEvent	Вызывается в индикаторах и экспертах при наступлении события ChartEvent для обработки изменений графика, вызванных действиями пользователя или работой MQL5-программ	Обработка событий
OnTester	Вызывается в экспертах при наступлении события Tester для выполнения необходимых действий по окончании тестирования	Обработка событий
OnTesterInit	Вызывается в экспертах при наступлении события TesterInit для выполнения необходимых действий перед началом оптимизации	Обработка событий
OnTesterDeinit	Вызывается в экспертах при наступлении события TesterDeinit для выполнения необходимых действий по окончании оптимизации эксперта	Обработка событий
OnTesterPass	Вызывается в экспертах при наступлении события TesterPass для обработки поступления нового фрейма	Обработка событий

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	данных во время оптимизации эксперта	
OrderCalcMargin	Вычисляет размер маржи, необходимой для указанного типа ордера, в валюте счета	Торговые функции
OrderCalcProfit	Вычисляет размер прибыли на основании переданных параметров в валюте счета	Торговые функции
OrderCheck	Проверяет достаточность средств для совершения требуемой торговой операции.	Торговые функции
OrderGetDouble	Возвращает запрошенное свойство ордера (double)	Торговые функции
OrderGetInteger	Возвращает запрошенное свойство ордера (datetime или int)	Торговые функции
OrderGetString	Возвращает запрошенное свойство ордера (string)	Торговые функции
OrderGetTicket	Возвращает тикет соответствующего ордера	Торговые функции
OrderSelect	Выбирает ордер для дальнейшей работы с ним	Торговые функции
OrderSend	Отправляет торговые запросы на сервер	Торговые функции
OrderSendAsync	Отправляет асинхронно торговые запросы без ожидания ответа торгового сервера	Торговые функции
OrdersTotal	Возвращает количество ордеров	Торговые функции
ParameterGetRange	Получает для input-переменной информацию о диапазоне значений и шаге изменения при оптимизации эксперта в тестере стратегий	Работа с результатами оптимизации
ParameterSetRange	Устанавливает правила использования input-переменной при оптимизации эксперта в тестере стратегий: значение, шаг изменения,	Работа с результатами оптимизации

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	начальное и конечное значения	
Period	Возвращает значение таймфрейма текущего графика	Проверка состояния
PeriodSeconds	Возвращает количество секунд в периоде	Общие функции
PlaySound	Воспроизводит звуковой файл	Общие функции
PlotIndexGetInteger	Возвращает значение свойства линии индикатора, имеющего целый тип	Пользовательские индикаторы
PlotIndexSetDouble	Задает значение свойства линии индикатора, имеющего тип double	Пользовательские индикаторы
PlotIndexSetInteger	Задает значение свойства линии индикатора, имеющего тип int	Пользовательские индикаторы
PlotIndexSetString	Задает значение свойства линии индикатора, имеющего тип string	Пользовательские индикаторы
Point	Возвращает размер пункта текущего инструмента в валюте котировки.	Проверка состояния
PositionGetDouble	Возвращает запрошенное свойство открытой позиции (double)	Торговые функции
PositionGetInteger	Возвращает запрошенное свойство открытой позиции (datetime или int)	Торговые функции
PositionGetString	Возвращает запрошенное свойство открытой позиции (string)	Торговые функции
PositionGetSymbol	Возвращает символ соответствующей открытой позиции	Торговые функции
PositionGetTicket	Возвращает тикет позиции по индексу в списке открытых позиций	Торговые функции
PositionSelect	Выбирает открытую позицию для дальнейшей работы с ней	Торговые функции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
PositionSelectByTicket	Выбирает открытую позицию для дальнейшей работы с ней по указанному тикету	Торговые функции
PositionsTotal	Возвращает количество открытых позиций	Торговые функции
pow	Возводит основание в указанную степень	Математические функции
Print	Выводит сообщение в журнал	Общие функции
PrintFormat	Форматирует и печатает наборы символов и значений в лог-файл в соответствие с заданным форматом	Общие функции
rand	Возвращает псевдослучайное целое число в диапазоне от 0 до 32767	Математические функции
ResetLastError	Устанавливает значение предопределенной переменной _LastError в ноль	Общие функции
ResourceCreate	Создает ресурс изображения на основе набора данных	Общие функции
ResourceFree	Удаляет динамически созданный ресурс (освобождает занятую ресурсом память)	Общие функции
ResourceReadImage	Читает данные графического ресурса, созданного функцией ResourceCreate() или сохраненного в EX5-файле при компиляции	Общие функции
ResourceSave	Сохраняет ресурс в указанный файл	Общие функции
round	Округляет число до ближайшего целого	Математические функции
SendFTP	Посылает файл по адресу, указанному в окне настроек на закладке "FTP"	Общие функции
SendMail	Посылает электронное письмо по адресу, указанному в окне настроек на закладке "Почта"	Общие функции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
SendNotification	Посылает Push-уведомления в мобильные терминалы, чьи MetaQuotes ID указаны на закладке "Уведомления"	Общие функции
SeriesInfoInteger	Возвращает информацию о состоянии исторических данных	Доступ к таймсериям и индикаторам
SetIndexBuffer	Связывает указанный индикаторный буфер с одномерным динамическим массивом типа double	Пользовательские индикаторы
ShortArrayToString	Копирует часть массива в строку	Преобразование данных
ShortToString	Преобразование код символа (unicode) в односимвольную строку	Преобразование данных
SignalBaseGetDouble	Возвращает значение свойства типа double для выбранного сигнала	Управление сигналами
SignalBaseGetInteger	Возвращает значение свойства типа integer для выбранного сигнала	Управление сигналами
SignalBaseGetString	Возвращает значение свойства типа string для выбранного сигнала	Управление сигналами
SignalBaseSelect	Выбирает для работы сигнал из базы торговых сигналов, доступных в терминале	Управление сигналами
SignalBaseTotal	Возвращает общее количество сигналов, доступных в терминале	Управление сигналами
SignalInfoGetDouble	Возвращает из настроек копирования торгового сигнала значение свойства типа double	Управление сигналами
SignalInfoGetInteger	Возвращает из настроек копирования торгового сигнала значение свойства типа integer	Управление сигналами
SignalInfoGetString	Возвращает из настроек копирования торгового сигнала значение свойства типа string	Управление сигналами

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
SignalInfoSetDouble	Устанавливает в настройках копирования торгового сигнала значение свойства типа double	Управление сигналами
SignalInfoSetInteger	Устанавливает в настройках копирования торгового сигнала значение свойства типа integer	Управление сигналами
SignalSubscribe	Производит подписку на копирование торгового сигнала	Управление сигналами
SignalUnsubscribe	Отменяет подписку на копирование торгового сигнала	Управление сигналами
sin	Возвращает синус числа	Математические функции
Sleep	Задерживает выполнение текущего эксперта или скрипта на определенный интервал	Общие функции
SocketCreate	Создает сокет с указанными флагами и возвращает его хэндл	Сетевые функции
SocketClose	Закрывает сокет	Сетевые функции
SocketConnect	Выполняет подключение к серверу с контролем таймаута	Сетевые функции
SocketIsConnected	Проверяет, подключен ли сокет в текущий момент времени	Сетевые функции
SocketIsReadable	Получает количество байт, которое можно прочитать из сокета	Сетевые функции
SocketIsWritable	Проверяет, возможна ли запись данных в сокет в текущий момент времени	Сетевые функции
SocketTimeouts	Устанавливает таймауты получения и отправки данных для системного объекта сокета	Сетевые функции
SocketRead	Читает данные из сокета	Сетевые функции
SocketSend	Записывает данные в сокете	Сетевые функции
SocketTlsHandshake	Инициирует защищенное TLS (SSL)-соединение с указанным хостом по протоколу TLS Handshake	Сетевые функции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
SocketTlsCertificate	Получает данные о сертификате, используемом для защиты сетевого соединения	Сетевые функции
SocketTlsRead	Читает данные из защищенного TLS-соединения	Сетевые функции
SocketTlsReadAvailable	Читает все доступные данные из защищенного TLS-соединения	Сетевые функции
SocketTlsSend	Отправляет данные через защищенное TLS-соединение	Сетевые функции
sqrt	Возвращает квадратный корень	Математические функции
srand	Устанавливает начальное состояние генератора псевдослучайных целых чисел	Математические функции
StringAdd	Присоединяет к концу строки по месту указанную подстроку	Строковые функции
StringBufferLen	Возвращает размер буфера, распределенного для строки	Строковые функции
StringCompare	Сравнивает две строки и возвращает 1, если первая строка больше второй; 0 - если строки равны; -1 (минус один) - если первая строка меньше второй	Строковые функции
StringConcatenate	Формирует строку из переданных параметров	Строковые функции
StringFill	Заполняет указанную строку по месту указанными символами	Строковые функции
StringFind	Поиск подстроки в строке	Строковые функции
StringFormat	Преобразует число в строку в соответствие с заданным форматом	Преобразование данных
StringGetCharacter	Возвращает значение символа, расположенного в указанной позиции строки	Строковые функции
StringInit	Инициализирует строку указанными символами и	Строковые функции

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	обеспечивает указанный размер строки	
StringLen	Возвращает число символов в строке	Строковые функции
StringReplace	Заменяет в строке все найденные подстроки на заданную последовательность символов.	Строковые функции
StringSetCharacter	Возвращает копию строки с измененным значением символа в указанной позиции	Строковые функции
StringSplit	Получает из указанной строки подстроки по заданному разделителю и возвращает количество полученных подстрок	Строковые функции
StringSubstr	Извлекает подстроку из текстовой строки, начинающейся с указанной позиции	Строковые функции
StringToCharArray	Посимвольно копирует преобразованную из Unicode в ANSI строку в указанное место массива типа uchar	Преобразование данных
StringToColor	Преобразует строку типа "R,G,B" или строку, содержащую наименование цвета, в значение типа color	Преобразование данных
StringToDouble	Преобразование строки, содержащей символьное представление числа, в число типа double	Преобразование данных
StringToInteger	Преобразование строки, содержащей символьное представление числа, в число типа int	Преобразование данных
StringToLower	Преобразует все символы указанной строки в строчные (маленькие) по месту	Строковые функции
StringToShortArray	Посимвольно копирует строку в указанное место массива типа ushort	Преобразование данных

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
StringToTime	Преобразование строки, содержащей время и/или дату в формате "yyyy.mm.dd [hh:mi]", в число типа datetime	Преобразование данных
StringToUpper	Преобразует все символы указанной строки в прописные (большие) по месту	Строковые функции
StringTrimLeft	Удаляет символы перевода каретки, пробелы и символы табуляции с начала строки	Строковые функции
StringTrimRight	Удаляет символы перевода каретки, пробелы и символы табуляции в конце строки	Строковые функции
StructToTime	Производит конвертацию из переменной типа структуры MqlDateTime в значение типа datetime	Дата и время
Symbol	Возвращает имя символа текущего графика.	Проверка состояния
SymbolInfoDouble	Возвращает значение типа double указанного символа для соответствующего свойства	Получение рыночной информации
SymbolInfoInteger	Возвращает значение целочисленного типа (long, datetime, int или bool) указанного символа для соответствующего свойства	Получение рыночной информации
SymbolInfoMarginRate	Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера	Получение рыночной информации
SymbolInfoSessionQuote	Позволяет получить время начала и время окончания указанной котировочной сессии для указанных символа и дня недели.	Получение рыночной информации
SymbolInfoSessionTrade	Позволяет получить время начала и время окончания указанной торговой сессии для указанных символа и дня недели.	Получение рыночной информации
SymbolInfoString	Возвращает значение типа string указанного символа для	Получение рыночной информации

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
	соответствующего свойства	
SymbolInfoTick	Возвращает текущие цены для указанного символа в переменной типа MqlTick	Получение рыночной информации
SymbolIsSynchronized	Проверяет синхронизированность данных по указанному символу в терминале с данными на торговом сервере	Получение рыночной информации
SymbolName	Возвращает наименование указанного символа	Получение рыночной информации
SymbolSelect	Выбирает символ в окне MarketWatch или убирает символ из окна	Получение рыночной информации
SymbolsTotal	Возвращает количество доступных (выбранных в MarketWatch или всех) символов	Получение рыночной информации
tan	Возвращает тангенс числа	Математические функции
TerminalClose	Посылает терминалу команду на завершение работы	Общие функции
TerminalInfoDouble	Возвращает значение типа double соответствующего свойства окружения mql5-программы	Проверка состояния
TerminalInfoInteger	Возвращает значение целого типа соответствующего свойства окружения mql5-программы	Проверка состояния
TerminalInfoString	Возвращает значение типа string соответствующего свойства окружения mql5-программы	Проверка состояния
TesterStatistics	Возвращает значение указанного статистического показателя, рассчитанного по результатам тестирования	Общие функции
TextGetSize	Возвращает ширину и высоту строки при текущих настройках шрифта	Графические объекты

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
TextOut	Выводит текст в пользовательский массив (буфер), предназначенный для создания графического ресурса	Графические объекты
TextSetFont	Устанавливает шрифт для вывода текста методами рисования (по умолчанию используется шрифт Arial 20)	Графические объекты
TimeCurrent	Возвращает последнее известное время сервера (время прихода последней котировки) в формате datetime	Дата и время
TimeDaylightSavings	Возвращает признак перехода на летнее /зимнее время	Дата и время
TimeGMT	Возвращает время GMT формате datetime с учетом перехода на зимнее или летнее время по локальному времени компьютера, на котором запущен клиентский терминал	Дата и время
TimeGMTOffset	Возвращает текущую разницу между временем GMT и локальным временем компьютера в секундах с учетом перехода на зимнее или летнее время	Дата и время
TimeLocal	Возвращает локальное компьютерное время в формате datetime	Дата и время
TimeToString	Преобразование значения, содержащего время в секундах, прошедшее с 01.01.1970, в строку формата "yyyy.mm.dd hh:mi"	Преобразование данных
TimeToStruct	Производит конвертацию из значения типа datetime в переменную типа структуры MqlDateTime	Дата и время
TimeTradeServer	Возвращает расчетное текущее время торгового сервера	Дата и время

© 2000-2025, MetaQuotes Ltd.

Функция	Действие	Раздел
UninitializeReason	Возвращает код причины деинициализации	Проверка состояния
WebRequest	Отправляет HTTP-запрос на указанный сервер	Общие функции
ZeroMemory	Обнуляет переменную, переданную по ссылке. Тип переменной может быть любым, исключение составляют только классы и структуры, имеющие конструкторы	Общие функции

© 2000-2025, MetaQuotes Ltd.

Список констант языка MQL5

Все константы MQL5 в алфавитном порядке.

Константа	Описание	Использование
__DATE__	Дата компиляции файла без времени (часы, минуты и секунды равны 0)	Print
__DATETIME__	Дата и время компиляции файла	Print
__FILE__	Имя текущего компилируемого файла	Print
__FUNCSIG__	Сигнатура функции, в теле которой расположен макрос. Вывод в лог полного описания функции с типами параметров может	Print

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	пригодиться при идентификации перегруженных функций	
__FUNCTION__	Имя функции, в теле которой расположен макрос	Print
__LINE__	Номер строки в исходном коде, на которой расположен данный макрос	Print
__MQLBUILD__, __MQL5BUILD__	Номер билда компилятора	Print
__PATH__	Абсолютный путь к текущему компилируемому файлу	Print
ACCOUNT_ASSETS	Текущий размер активов на счёте	AccountInfoDouble
ACCOUNT_BALANCE	Баланс счета в валюте	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	депозита	
ACCOUNT_COMMISSION_BLOCKED	Текущая сумма заблокированных комиссий по счёту	AccountInfoDouble
ACCOUNT_COMPANY	Имя компании, обслуживающей счет	AccountInfoString
ACCOUNT_CREDIT	Размер предоставленного кредита в валюте депозита	AccountInfoDouble
ACCOUNT_CURRENCY	Валюта депозита	AccountInfoString
ACCOUNT_EQUITY	Значение собственных средств на счете в валюте депозита	AccountInfoDouble
ACCOUNT_LEVERAGE	Размер предоставленного плеча	AccountInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ACCOUNT_LIABILITIES	Текущий размер обязательств на счёте	AccountInfoDouble
ACCOUNT_LIMIT_ORDERS	Максимально допустимое количество действующих отложенных ордеров	AccountInfoInteger
ACCOUNT_LOGIN	Номер счета	AccountInfoInteger
ACCOUNT_MARGIN	Размер зарезервированных залоговых средств на счете в валюте депозита	AccountInfoDouble
ACCOUNT_MARGIN_FREE	Размер свободных средств на счете в валюте депозита, доступных для открытия позиции	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ACCOUNT_MARGIN_INITIAL	Размер средств, зарезервированных на счёте, для обеспечения гарантийной суммы по всем отложенным ордерам	AccountInfoDouble
ACCOUNT_MARGIN_LEVEL	Уровень залоговых средств на счете в процентах	AccountInfoDouble
ACCOUNT_MARGIN_MAINTENANCE	Размер средств, зарезервированных на счёте, для обеспечения минимальной суммы по всем открытым позициям	AccountInfoDouble
ACCOUNT_MARGIN_SO_CALL	Уровень залоговых средств,	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	при котором требуется пополнение счета (Margin Call). В зависимости от установленного ACCOUNT_MARGIN_SO_MODE выражается в процентах либо в валюте депозита	
ACCOUNT_MARGIN_SO_MODE	Режим задания минимально допустимого уровня залоговых средств	AccountInfoInteger
ACCOUNT_MARGIN_SO_SO	Уровень залоговых средств, при достижении которого происходит	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	принудительное закрытие самой убыточной позиции (Stop Out). В зависимости от установленного ACCOUNT_MARGIN_SO_MODE выражается в процентах либо в валюте депозита	
ACCOUNT_NAME	Имя клиента	AccountInfoString
ACCOUNT_PROFIT	Размер текущей прибыли на счете в валюте депозита	AccountInfoDouble
ACCOUNT_SERVER	Имя торгового сервера	AccountInfoString
ACCOUNT_STOPOUT_MODE_MONEY	Уровень задается в деньгах	AccountInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ACCOUNT_STOPOUT_MODE_PERCENT	Уровень задается в процентах	AccountInfoInteger
ACCOUNT_TRADE_ALLOWED	Разрешенность торговли для текущего счета	AccountInfoInteger
ACCOUNT_TRADE_EXPERT	Разрешенность торговли для эксперта	AccountInfoInteger
ACCOUNT_TRADE_MODE	Тип торгового счета	AccountInfoInteger
ACCOUNT_TRADE_MODE_CONTEST	Конкурсный торговый счет	AccountInfoInteger
ACCOUNT_TRADE_MODE_DEMO	Демонстрационный торговый счет	AccountInfoInteger
ACCOUNT_TRADE_MODE_REAL	Реальный торговый счет	AccountInfoInteger
ALIGN_CENTER	Выравнивание по центру (только для объекта "Поле ввода")	ObjectSetInteger, ObjectGetInteger, ChartScreenShot

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ALIGN_LEFT	Выравнивание по левой границе	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_RIGHT	Выравнивание по правой границе	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ANCHOR_CENTER	Точка привязки строго по центру объекта	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT	Точка привязки слева по центру	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_LOWER	Точка привязки в левом нижнем углу	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_UPPER	Точка привязки в левом верхнем углу	ObjectSetInteger, ObjectGetInteger
ANCHOR_LOWER	Точка привязки снизу по центру	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT	Точка привязки справа по центру	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ANCHOR_RIGHT_LOWER	Точка привязки в правом нижнем углу	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_UPPER	Точка привязки в правом верхнем углу	ObjectSetInteger, ObjectGetInteger
ANCHOR_UPPER	Точка привязки сверху по центру	ObjectSetInteger, ObjectGetInteger
BASE_LINE	Основная линия	Линии индикаторов
BOOK_TYPE_BUY	Заявка на покупку	MqlBookInfo
BOOK_TYPE_BUY_MARKET	Заявка на покупку по рыночной цене	MqlBookInfo
BOOK_TYPE_SELL	Заявка на продажу	MqlBookInfo
BOOK_TYPE_SELL_MARKET	Заявка на продажу по рыночной цене	MqlBookInfo
BORDER_FLAT	Плоский вид	ObjectSetInteger, ObjectGetInteger
BORDER_RAISED	Выпуклый вид	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
BORDER_SUNKEN	Вогнутый вид	ObjectSetInteger, ObjectGetInteger
CHAR_MAX	Максимальное значение, которое может быть представлено типом char	Константы числовых типов
CHAR_MIN	Минимальное значение, которое может быть представлено типом char	Константы числовых типов
CHART_AUTOSCROLL	Режим автоматического перехода к правому краю графика	ChartSetInteger, ChartGetInteger
CHART_BARS	Отображение в виде баров	ChartSetInteger
CHART_BEGIN	Начало графика (самые старые цены)	ChartNavigate
CHART_BRING_TO_TOP	Показ графика	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	поверх всех других	
CHART_CANDLES	Отображение в виде японских свечей	ChartSetInteger
CHART_COLOR_ASK	Цвет линии Ask-цены	ChartSetInteger, ChartGetInteger
CHART_COLOR_BACKGROUND	Цвет фона графика	ChartSetInteger, ChartGetInteger
CHART_COLOR_BID	Цвет линии Bid-цены	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BEAR	Цвет тела медвежьей свечи	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BULL	Цвет тела бычьей свечи	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_DOWN	Цвет бара вниз, тени и окантовки тела медвежьей свечи	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_LINE	Цвет линии графика и японски	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	х свечей "Доджи"	
CHART_COLOR_CHART_UP	Цвет бара вверх, тени и окантовки тела бычьей свечи	ChartSetInteger, ChartGetInteger
CHART_COLOR_FOREGROUND	Цвет осей, шкалы и строки OHLC	ChartSetInteger, ChartGetInteger
CHART_COLOR_GRID	Цвет сетки	ChartSetInteger, ChartGetInteger
CHART_COLOR_LAST	Цвет линии цены последней совершенной сделки (Last)	ChartSetInteger, ChartGetInteger
CHART_COLOR_STOP_LEVEL	Цвет уровней стоп-ордеров (Stop Loss и Take Profit)	ChartSetInteger, ChartGetInteger
CHART_COLOR_VOLUME	Цвет объемов и уровней открытия позиций	ChartSetInteger, ChartGetInteger
CHART_COMMENT	Текст коммент	ChartSetString, ChartGetString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ария на графике	
CHART_CURRENT_POS	Текущая позиция	ChartNavigate
CHART_DRAG_TRADE_LEVELS	Разрешение на перетаскивание торговых уровней на графике с помощью мышки. По умолчанию режим перетаскивания включен (значение true)	ChartSetInteger, ChartGetInteger
CHART_EVENT_MOUSE_MOVE	Отправка всем mql5-программам на графике сообщений о событиях перемещения и нажатия кнопок мыши (CHART_EVENT_MOUSE_MOVE)	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
CHART_EVENT_OBJECT_CREATE	Отправка всем mql5-программам на графике сообщений о событии создания графического объекта (CHART_EVENT_OBJECT_CREATE)	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_DELETE	Отправка всем mql5-программам на графике сообщений о событии уничтожения графического объекта (CHART_EVENT_OBJECT_DELETE)	ChartSetInteger, ChartGetInteger
CHART_FIRST_VISIBLE_BAR	Номер первого видимого бара на графике. Индексация баров	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	соответствует таймсерии.	
CHART_FIXED_MAX	Фиксированный максимум графика	ChartSetDouble, ChartGetDouble
CHART_FIXED_MIN	Фиксированный минимум графика	ChartSetDouble, ChartGetDouble
CHART_FIXED_POSITION	Положение фиксированной позиции графика от левого края в процентах. Фиксированная позиция графика обозначена маленьким серым треугольником на горизонтальной оси времени и показывается только в том	ChartSetDouble, ChartGetDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	случае, если отключена автоматическая прокрутка к правому краю при поступлении нового тика (смотри свойство CHART_AUTOSCROLL). Бар, который находится на фиксированной позиции, остаётся на том же месте при увеличении и уменьшении масштаба.	
CHART_FOREGROUND	Ценовой график на переднем плане	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
CHART_HEIGHT_IN_PIXELS	Высота графика в пикселях	ChartSetInteger, ChartGetInteger
CHART_IS_OBJECT	Признак для идентификации объекта "График" (OBJ_CHART) - для графического объекта возвращает true. Для настоящего графика значение равно false	ChartSetInteger, ChartGetInteger
CHART_LINE	Отображение в виде линии, проведенной по ценам Close	ChartSetInteger
CHART_MODE	Тип графика (свечи, бары или линия)	ChartSetInteger, ChartGetInteger
CHART_MOUSE_SCROLL	Прокрутка графика левой	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	кнопкой мышки по горизонтали. Прокрутка по вертикали также будет доступна, если установлено в true значение любого из трех свойств: CHART_SCALEFIX, CHART_SCALEFIX_11 или CHART_SCALE_PT_PER_BAR	
CHART_POINTS_PER_BAR	Значение масштаба в пунктах на бар	ChartSetDouble, ChartGetDouble
CHART_PRICE_MAX	Максимум графика	ChartSetDouble, ChartGetDouble
CHART_PRICE_MIN	Минимум графика	ChartSetDouble, ChartGetDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
CHART_SCALE	Масштаб	ChartSetInteger, ChartGetInteger
CHART_SCALE_PT_PER_BAR	Режим указания масштаба в пунктах на бар	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX	Режим фиксированного масштаба	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX_11	Режим масштаба 1:1	ChartSetInteger, ChartGetInteger
CHART_SHIFT	Режим отступа ценового графика от правого края	ChartSetInteger, ChartGetInteger
CHART_SHIFT_SIZE	Размер отступа нулевого бара от правого края в процентах	ChartSetDouble, ChartGetDouble
CHART_SHOW_ASK_LINE	Отображение значения Ask горизонтальной линией на графике	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
CHART_SHOW_BID_LINE	Отображение значения Bid горизонтальной линией на графике	ChartSetInteger, ChartGetInteger
CHART_SHOW_DATE_SCALE	Отображение на графике шкалы времени	ChartSetInteger, ChartGetInteger
CHART_SHOW_GRID	Отображение сетки на графике	ChartSetInteger, ChartGetInteger
CHART_SHOW_LAST_LINE	Отображение значения Last горизонтальной линией на графике	ChartSetInteger, ChartGetInteger
CHART_SHOW_OBJECT_DESCR	Всплывающие описания графических объектов	ChartSetInteger, ChartGetInteger
CHART_SHOW_OHLC	Отображение в левом верхнем углу значений OHLC	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
CHART_SHOW_ONE_CLICK	Отображение на графике панели быстрой торговли (опция "Торговля одним кликом")	ChartSetInteger, ChartGetInteger
CHART_SHOW_PERIOD_SEP	Отображение вертикальных разделителей между соседними периодами	ChartSetInteger, ChartGetInteger
CHART_SHOW_PRICE_SCALE	Отображение на графике ценовой шкалы	ChartSetInteger, ChartGetInteger
CHART_SHOW_TRADE_LEVELS	Отображение на графике торговых уровней (уровни открытых позиций, Stop Loss, Take Profit и отложен	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ных ордеров)	
CHART_SHOW_VOLUMES	Отображение объемов на графике	ChartSetInteger, ChartGetInteger
CHART_VISIBLE_BARS	Количество баров на графике, доступных для отображения	ChartSetInteger, ChartGetInteger
CHART_VOLUME_HIDE	Объемы не показаны	ChartSetInteger
CHART_VOLUME_REAL	Торговые объемы	ChartSetInteger
CHART_VOLUME_TICK	Тиковые объемы	ChartSetInteger
CHART_WIDTH_IN_BARS	Ширина графика в барах	ChartSetInteger, ChartGetInteger
CHART_WIDTH_IN_PIXELS	Ширина графика в пикселях	ChartSetInteger, ChartGetInteger
CHART_WINDOW_HANDLE	Хэндл графика (HWND)	ChartSetInteger, ChartGetInteger
CHART_WINDOW_IS_VISIBLE	Видимость подокон	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
CHART_WINDOW_YDISTANCE	Дистанция в пикселях по вертикальной оси Y между верхней рамкой подокна индикатора и верхней рамкой главного окна графика. При наступлении событий мыши координаты курсора передаются в координатах главного окна графика, в то время как координаты графических объектов в подокне индикатора задаются относит	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ельно верхнего левого угла подокна. Значение требуется для перевода абсолютных координат главного графика в локальные координаты подокна для корректной работы с графическими объектами, у которых координаты задаются относительно верхнего левого угла рамки подокна.	

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
CHART_WINDOWS_TOTAL	Общее количество окон графика, включая подокна индикаторов	ChartSetInteger, ChartGetInteger
CHARTEVENT_CHART_CHANGE	Изменение размеров графика или изменение свойств графика через диалог свойств	OnChartEvent
CHARTEVENT_CLICK	Нажатие мышки на графике	OnChartEvent
CHARTEVENT_CUSTOM	Начальный номер события из диапазона пользовательских событий	OnChartEvent
CHARTEVENT_CUSTOM_LAST	Конечный номер события из диапазона пользов	OnChartEvent

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ательских событий	
CHARTEVENT_KEYDOWN	Нажатие клавиатуры	OnChartEvent
CHARTEVENT_MOUSE_MOVE	Перемещение мыши и нажатие кнопок мыши (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE=true)	OnChartEvent
CHARTEVENT_OBJECT_CHANGE	Изменение свойств графического объекта через диалог свойств	OnChartEvent
CHARTEVENT_OBJECT_CLICK	Нажатие мышки на графическом объекте	OnChartEvent
CHARTEVENT_OBJECT_CREATE	Создание графического	OnChartEvent

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	объекта (если для графика установлено свойство CHART_EVENT_OBJECT_CREATE =true)	
CHARTEVENT_OBJECT_DELETE	Удаление графического объекта (если для графика установлено свойство CHART_EVENT_OBJECT_DELETE= true)	OnChartEvent
CHARTEVENT_OBJECT_DRAG	Перетаскивание графического объекта	OnChartEvent
CHARTEVENT_OBJECT_ENDEDIT	Окончание редактирования текста в графическом объекте Edit	OnChartEvent

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
CHARTS_MAX	Максимально возможное количество одновременно открытых графиков в терминале	Прочие константы
CHIKOUSPAN_LINE	Линия Chikou Span	Линии индикаторов
clrAliceBlue	Alice Blue	Набор Web-цветов
clrAntiqueWhite	Antique White	Набор Web-цветов
clrAqua	Aqua	Набор Web-цветов
clrAquamarine	Aquamarine	Набор Web-цветов
clrBeige	Beige	Набор Web-цветов
clrBisque	Bisque	Набор Web-цветов
clrBlack	Black	Набор Web-цветов
clrBlanchedAlmond	Blanched Almond	Набор Web-цветов
clrBlue	Blue	Набор Web-цветов
clrBlueViolet	Blue Violet	Набор Web-цветов
clrBrown	Brown	Набор Web-цветов
clrBurlyWood	Burly Wood	Набор Web-цветов
clrCadetBlue	Cadet Blue	Набор Web-цветов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
clrChartreuse	Chartreuse	Набор Web-цветов
clrChocolate	Chocolate	Набор Web-цветов
clrCoral	Coral	Набор Web-цветов
clrCornflowerBlue	Cornflower Blue	Набор Web-цветов
clrCornsilk	Cornsilk	Набор Web-цветов
clrCrimson	Crimson	Набор Web-цветов
clrDarkBlue	Dark Blue	Набор Web-цветов
clrDarkGoldenrod	Dark Goldenrod	Набор Web-цветов
clrDarkGray	Dark Gray	Набор Web-цветов
clrDarkGreen	Dark Green	Набор Web-цветов
clrDarkKhaki	Dark Khaki	Набор Web-цветов
clrDarkOliveGreen	Dark Olive Green	Набор Web-цветов
clrDarkOrange	Dark Orange	Набор Web-цветов
clrDarkOrchid	Dark Orchid	Набор Web-цветов
clrDarkSalmon	Dark Salmon	Набор Web-цветов
clrDarkSeaGreen	Dark Sea Green	Набор Web-цветов
clrDarkSlateBlue	Dark Slate Blue	Набор Web-цветов
clrDarkSlateGray	Dark Slate Gray	Набор Web-цветов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
clrDarkTurquoise	Dark Turquoise	Набор Web-цветов
clrDarkViolet	Dark Violet	Набор Web-цветов
clrDeepPink	Deep Pink	Набор Web-цветов
clrDeepSkyBlue	Deep Sky Blue	Набор Web-цветов
clrDimGray	Dim Gray	Набор Web-цветов
clrDodgerBlue	Dodger Blue	Набор Web-цветов
clrFireBrick	Fire Brick	Набор Web-цветов
clrForestGreen	Forest Green	Набор Web-цветов
clrGainsboro	Gainsboro	Набор Web-цветов
clrGold	Gold	Набор Web-цветов
clrGoldenrod	Goldenrod	Набор Web-цветов
clrGray	Gray	Набор Web-цветов
clrGreen	Green	Набор Web-цветов
clrGreenYellow	Green Yellow	Набор Web-цветов
clrHoneydew	Honeydew	Набор Web-цветов
clrHotPink	Hot Pink	Набор Web-цветов
clrIndianRed	Indian Red	Набор Web-цветов
clrIndigo	Indigo	Набор Web-цветов
clrIvory	Ivory	Набор Web-цветов
clrKhaki	Khaki	Набор Web-цветов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
clrLavender	Lavender	Набор Web-цветов
clrLavenderBlush	Lavender Blush	Набор Web-цветов
clrLawnGreen	Lawn Green	Набор Web-цветов
clrLemonChiffon	Lemon Chiffon	Набор Web-цветов
clrLightBlue	Light Blue	Набор Web-цветов
clrLightCoral	Light Coral	Набор Web-цветов
clrLightCyan	Light Cyan	Набор Web-цветов
clrLightGoldenrod	Light Goldenrod	Набор Web-цветов
clrLightGray	Light Gray	Набор Web-цветов
clrLightGreen	Light Green	Набор Web-цветов
clrLightPink	Light Pink	Набор Web-цветов
clrLightSalmon	Light Salmon	Набор Web-цветов
clrLightSeaGreen	Light Sea Green	Набор Web-цветов
clrLightSkyBlue	Light Sky Blue	Набор Web-цветов
clrLightSlateGray	Light Slate Gray	Набор Web-цветов
clrLightSteelBlue	Light Steel Blue	Набор Web-цветов
clrLightYellow	Light Yellow	Набор Web-цветов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
clrLime	Lime	Набор Web-цветов
clrLimeGreen	Lime Green	Набор Web-цветов
clrLinen	Linen	Набор Web-цветов
clrMagenta	Magenta	Набор Web-цветов
clrMaroon	Maroon	Набор Web-цветов
clrMediumAquamarine	Medium Aquamarine	Набор Web-цветов
clrMediumBlue	Medium Blue	Набор Web-цветов
clrMediumOrchid	Medium Orchid	Набор Web-цветов
clrMediumPurple	Medium Purple	Набор Web-цветов
clrMediumSeaGreen	Medium Sea Green	Набор Web-цветов
clrMediumSlateBlue	Medium Slate Blue	Набор Web-цветов
clrMediumSpringGreen	Medium Spring Green	Набор Web-цветов
clrMediumTurquoise	Medium Turquoise	Набор Web-цветов
clrMediumVioletRed	Medium Violet Red	Набор Web-цветов
clrMidnightBlue	Midnight Blue	Набор Web-цветов
clrMintCream	Mint Cream	Набор Web-цветов
clrMistyRose	Misty Rose	Набор Web-цветов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
clrMoccasin	Moccasin	Набор Web-цветов
clrNavajoWhite	Navajo White	Набор Web-цветов
clrNavy	Navy	Набор Web-цветов
clrNONE	Отсутствие цвета	Прочие константы
clrOldLace	Old Lace	Набор Web-цветов
clrOlive	Olive	Набор Web-цветов
clrOliveDrab	Olive Drab	Набор Web-цветов
clrOrange	Orange	Набор Web-цветов
clrOrangeRed	Orange Red	Набор Web-цветов
clrOrchid	Orchid	Набор Web-цветов
clrPaleGoldenrod	Pale Goldenrod	Набор Web-цветов
clrPaleGreen	Pale Green	Набор Web-цветов
clrPaleTurquoise	Pale Turquoise	Набор Web-цветов
clrPaleVioletRed	Pale Violet Red	Набор Web-цветов
clrPapayaWhip	Papaya Whip	Набор Web-цветов
clrPeachPuff	Peach Puff	Набор Web-цветов
clrPeru	Peru	Набор Web-цветов
clrPink	Pink	Набор Web-цветов
clrPlum	Plum	Набор Web-цветов
clrPowderBlue	Powder Blue	Набор Web-цветов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
clrPurple	Purple	Набор Web-цветов
clrRed	Red	Набор Web-цветов
clrRosyBrown	Rosy Brown	Набор Web-цветов
clrRoyalBlue	Royal Blue	Набор Web-цветов
clrSaddleBrown	Saddle Brown	Набор Web-цветов
clrSalmon	Salmon	Набор Web-цветов
clrSandyBrown	Sandy Brown	Набор Web-цветов
clrSeaGreen	Sea Green	Набор Web-цветов
clrSeashell	Seashell	Набор Web-цветов
clrSienna	Sienna	Набор Web-цветов
clrSilver	Silver	Набор Web-цветов
clrSkyBlue	Sky Blue	Набор Web-цветов
clrSlateBlue	Slate Blue	Набор Web-цветов
clrSlateGray	Slate Gray	Набор Web-цветов
clrSnow	Snow	Набор Web-цветов
clrSpringGreen	Spring Green	Набор Web-цветов
clrSteelBlue	Steel Blue	Набор Web-цветов
clrTan	Tan	Набор Web-цветов
clrTeal	Teal	Набор Web-цветов
clrThistle	Thistle	Набор Web-цветов
clrTomato	Tomato	Набор Web-цветов
clrTurquoise	Turquoise	Набор Web-цветов
clrViolet	Violet	Набор Web-цветов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
clrWheat	Wheat	Набор Web-цветов
clrWhite	White	Набор Web-цветов
clrWhiteSmoke	White Smoke	Набор Web-цветов
clrYellow	Yellow	Набор Web-цветов
clrYellowGreen	Yellow Green	Набор Web-цветов
CORNER_LEFT_LOWER	Центр координат в левом нижнем углу графика	ObjectSetInteger, ObjectGetInteger
CORNER_LEFT_UPPER	Центр координат в левом верхнем углу графика	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_LOWER	Центр координат в правом нижнем углу графика	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_UPPER	Центр координат в правом верхнем углу графика	ObjectSetInteger, ObjectGetInteger
CP_ACP	Текущая кодовая страница ANSI кодировка в операци	CharArrayToString, StringToCharArray, FileOpen

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	онной системе Windows	
CP_MACCP	Текущая кодовая страница Macintosh. Примечание: Это значение преимущественно используется в ранее созданных программных кодах и теперь в нем нет необходимости, так как современные компьютеры Macintosh используют Unicode кодировку.	CharArrayToString, StringToCharArray, FileOpen
CP_OEMCP	Текущая кодовая страница OEM.	CharArrayToString, StringToCharArray, FileOpen
CP_SYMBOL	Кодовая страниц	CharArrayToString, StringToCharArray, FileOpen

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	а Symbol	
CP_THREAD_ACP	Кодировка Windows ANSI для текущего потока выполнения.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF7	Кодовая страница UTF-7.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF8	Кодовая страница UTF-8.	CharArrayToString, StringToCharArray, FileOpen
CRYPT_AES128	Шифрование AES с ключом 128 бит (16 байт)	CryptEncode, CryptDecode
CRYPT_AES256	Шифрование AES с ключом 256 бит (32 байта)	CryptEncode, CryptDecode
CRYPT_ARCH_ZIP	ZIP архивирование	CryptEncode, CryptDecode
CRYPT_BASE64	Шифрование BASE64 (перекодировка)	CryptEncode, CryptDecode
CRYPT_DES	Шифрование DES с ключом	CryptEncode, CryptDecode

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	56 бит (7 байт)	
CRYPT_HASH_MD5	Расчёт HASH MD5	CryptEncode, CryptDecode
CRYPT_HASH_SHA1	Расчёт HASH SHA1	CryptEncode, CryptDecode
CRYPT_HASH_SHA256	Расчёт HASH SHA256	CryptEncode, CryptDecode
DBL_DIG	Количество значимых десятичных знаков	Константы числовых типов
DBL_EPSILON	Наименьшее число для которого выполняется условие 1.0+DBL_EPSILON != 1.0	Константы числовых типов
DBL_MANT_DIG	Количество битов в мантиссе	Константы числовых типов
DBL_MAX	Максимальное значение, которое может быть предста	Константы числовых типов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	влено типом double	
DBL_MAX_10_EXP	Максимальное десятичное значение степени экспоненты	Константы числовых типов
DBL_MAX_EXP	Максимальное двоичное значение степени экспоненты	Константы числовых типов
DBL_MIN	Минимальное положительное значение, которое может быть представлено типом double	Константы числовых типов
DBL_MIN_10_EXP	Минимальное десятичное значение степени экспоненты	Константы числовых типов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
DBL_MIN_EXP	Минимальное двоичное значение степени экспоненты	Константы числовых типов
DEAL_COMMENT	Комментарий к сделке	HistoryDealGetString
DEAL_COMMISSION	Комиссия по сделке	HistoryDealGetDouble
DEAL_ENTRY	Направление сделки - вход в рынок, выход из рынка или разворот	HistoryDealGetInteger
DEAL_ENTRY_IN	Вход в рынок	HistoryDealGetInteger
DEAL_ENTRY_INOUT	Разворот	HistoryDealGetInteger
DEAL_ENTRY_OUT	Выход из рынка	HistoryDealGetInteger
DEAL_MAGIC	Magic number для сделки (смотри ORDER_MAGIC)	HistoryDealGetInteger
DEAL_ORDER	Ордер, на основан	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ие которого выполнена сделка	
DEAL_POSITION_ID	Идентификатор позиции, в открытии, изменении или закрытии которой участвовала эта сделка. Каждая позиция имеет уникальный идентификатор, который присваивается всем сделкам, совершенным на инструменте в течение всей жизни позиции.	HistoryDealGetInteger
DEAL_PRICE	Цена сделки	HistoryDealGetDouble
DEAL_PROFIT	Финансовый	HistoryDealGetDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	результат сделки	
DEAL_SWAP	Накопленный своп при закрытии	HistoryDealGetDouble
DEAL_SYMBOL	Имя символа, по которому произведена сделка	HistoryDealGetString
DEAL_TIME	Время совершения сделки	HistoryDealGetInteger
DEAL_TIME_MSC	Время совершения сделки в миллисекундах с 01.01.1970	HistoryDealGetInteger
DEAL_TYPE	Тип сделки	HistoryDealGetInteger
DEAL_TYPE_BALANCE	Начисление баланса	HistoryDealGetInteger
DEAL_TYPE_BONUS	Перечисление бонусов	HistoryDealGetInteger
DEAL_TYPE_BUY	Покупка	HistoryDealGetInteger
DEAL_TYPE_BUY_CANCELED	Отменная сделка покупки	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	. Возможная ситуация, когда ранее совершенная сделка на покупку отменяется. В таком случае тип ранее совершенной сделки (DEAL_TYPE_BUY) меняется на DEAL_TYPE_BUY_CANCELED, а ее прибыль/убыток обнуляется. Ранее полученная прибыль/убыток начисляется/списывается со счета отдельной балансовой	

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	операций	
DEAL_TYPE_CHARGE	Дополнительные сборы	HistoryDealGetInteger
DEAL_TYPE_COMMISSION	Дополнительные комиссии	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_DAILY	Агентская комиссия, начисляемая в конце торгового дня	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Агентская комиссия, начисляемая в конце месяца	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_DAILY	Комиссия, начисляемая в конце торгового дня	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_MONTHLY	Комиссия, начисляемая в конце месяца	HistoryDealGetInteger
DEAL_TYPE_CORRECTION	Корректирующая запись	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
DEAL_TYPE_CREDIT	Начисление кредита	HistoryDealGetInteger
DEAL_TYPE_INTEREST	Начисления процентов на свободные средства	HistoryDealGetInteger
DEAL_TYPE_SELL	Продажа	HistoryDealGetInteger
DEAL_TYPE_SELL_CANCELED	Отменная сделка продажи. Возможная ситуация, когда ранее совершенная сделка на продажу отменяется. В таком случае тип ранее совершенной сделки (DEAL_TYPE_SELL) меняется на DEAL_TYPE_SELL_CANCELED, а	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ее прибыль/убыток обнуляется. Ранее полученная прибыль/убыток начисляется/списывается со счета отдельной балансовой операцией	
DEAL_VOLUME	Объем сделки	HistoryDealGetDouble
DRAW_ARROW	Отрисовка стрелками	Стили рисования
DRAW_BARS	Отображение в виде баров	Стили рисования
DRAW_CANDLES	Отображение в виде свечей	Стили рисования
DRAW_COLOR_ARROW	Отрисовка разноцветными стрелками	Стили рисования
DRAW_COLOR_BARS	Разноцветные бары	Стили рисования

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
DRAW_COLOR_CANDLES	Разноцветные свечи	Стили рисования
DRAW_COLOR_HISTOGRAM	Разноцветная гистограмма от нулевой линии	Стили рисования
DRAW_COLOR_HISTOGRAM2	Разноцветная гистограмма на двух индикаторных буферах	Стили рисования
DRAW_COLOR_LINE	Разноцветная линия	Стили рисования
DRAW_COLOR_SECTION	Разноцветные отрезки	Стили рисования
DRAW_COLOR_ZIGZAG	Разноцветный ZigZag	Стили рисования
DRAW_FILLING	Цветовая заливка между двумя уровнями	Стили рисования
DRAW_HISTOGRAM	Гистограмма от нулевой линии	Стили рисования
DRAW_HISTOGRAM2	Гистограмма на двух индикаторных буферах	Стили рисования

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
DRAW_LINE	Линия	Стили рисования
DRAW_NONE	Не отрисовывается	Стили рисования
DRAW_SECTION	Отрезки	Стили рисования
DRAW_ZIGZAG	Стиль Zigzag допускает вертикальные отрезки на баре	Стили рисования
ELLIOTT_CYCLE	Цикл (Cycle)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_GRAND_SUPERCYCLE	Главный Суперцикл (Grand Supercycle)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_INTERMEDIATE	Промежуточное звено (Intermediate)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINOR	Второстепенный цикл (Minor)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUETTE	Секунда (Minuette)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUTE	Минута (Minute)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_PRIMARY	Первичный цикл (Primary)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUBMINUETTE	Субсекунда	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	(Subminuette)	
ELLIOTT_SUPERCYCLE	Суперцикл (Supercycle)	ObjectSetInteger, ObjectGetInteger
EMPTY_VALUE	Пустое значение в индикаторном буфере	Прочие константы
ERR_ACCOUNT_WRONG_PROPERTY	Ошибочный идентификатор свойства счета	GetLastError
ERR_ARRAY_BAD_SIZE	Запрашиваемый размер массива превышает 2 гигабайта	GetLastError
ERR_ARRAY_RESIZE_ERROR	Недостаточно памяти для перераспределения массива либо попытка изменения размера статического массива	GetLastError
ERR_BOOKS_CANNOT_ADD	Стакан цен не	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	может быть добавлен	
ERR_BOOKS_CANNOT_DELETE	Стакан цен не может быть удален	GetLastError
ERR_BOOKS_CANNOT_GET	Данные стакана цен не могут быть получены	GetLastError
ERR_BOOKS_CANNOT_SUBSCRIBE	Ошибка при подписке на получение новых данных стакана цен	GetLastError
ERR_BUFFERS_NO_MEMORY	Недостаточно памяти для распределения индикаторных буферов	GetLastError
ERR_BUFFERS_WRONG_INDEX	Ошибочный индекс своего индикаторного буфера	GetLastError
ERR_CANNOT_CLEAN_DIRECTORY	Не удалось	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	очистить директорию (возможно, один или несколько файлов заблокированы и операция удаления не удалась)	
ERR_CANNOT_DELETE_DIRECTORY	Директория не может быть удалена	GetLastError
ERR_CANNOT_DELETE_FILE	Ошибка удаления файла	GetLastError
ERR_CANNOT_OPEN_FILE	Ошибка открытия файла	GetLastError
ERR_CHAR_ARRAY_ONLY	Должен быть массив типа char	GetLastError
ERR_CHART_CANNOT_CHANGE	Ошибка при изменении для графика символа и периода	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_CHART_CANNOT_CREATE_TIMER	Ошибка при создании таймера	GetLastError
ERR_CHART_CANNOT_OPEN	Ошибка открытия графика	GetLastError
ERR_CHART_INDICATOR_CANNOT_ADD	Ошибка при добавлении индикатора на график	GetLastError
ERR_CHART_INDICATOR_CANNOT_DEL	Ошибка при удалении индикатора с графика	GetLastError
ERR_CHART_INDICATOR_NOT_FOUND	Индикатор не найден на указанном графике	GetLastError
ERR_CHART_NAVIGATE_FAILED	Ошибка навигации по графику	GetLastError
ERR_CHART_NO_EXPERT	У графика нет эксперта, который мог бы обработать событие	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_CHART_NO_REPLY	График не отвечает	GetLastError
ERR_CHART_NOT_FOUND	График не найден	GetLastError
ERR_CHART_SCREENSHOT_FAILED	Ошибка при создании скриншота	GetLastError
ERR_CHART_TEMPLATE_FAILED	Ошибка при применении шаблона	GetLastError
ERR_CHART_WINDOW_NOT_FOUND	Подокно, содержащее указанный индикатор, не найдено	GetLastError
ERR_CHART_WRONG_ID	Ошибочный идентификатор графика	GetLastError
ERR_CHART_WRONG_PARAMETER	Ошибочное значение параметра для функции по работе с графиком	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_CHART_WRONG_PROPERTY	Ошибочный идентификатор свойства графика	GetLastError
ERR_CUSTOM_WRONG_PROPERTY	Ошибочный идентификатор свойства пользовательского индикатора	GetLastError
ERR_DIRECTORY_NOT_EXIST	Директория не существует	GetLastError
ERR_DOUBLE_ARRAY_ONLY	Должен быть массив типа double	GetLastError
ERR_FILE_BINSTRINGSIZE	Должен быть указан размер строки, так как файл открыт как бинарный	GetLastError
ERR_FILE_CACHEBUFFER_ERROR	Недостаточно памяти для кеша чтения	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_FILE_CANNOT_REWRITE	Файл не может быть переписан	GetLastError
ERR_FILE_IS_DIRECTORY	Это не файл, а директория	GetLastError
ERR_FILE_ISNOT_DIRECTORY	Это файл, а не директория	GetLastError
ERR_FILE_NOT_EXIST	Файл не существует	GetLastError
ERR_FILE_NOTBIN	Файл должен быть открыт как бинарный	GetLastError
ERR_FILE_NOTCSV	Файл должен быть открыт как CSV	GetLastError
ERR_FILE_NOTTOREAD	Файл должен быть открыт для чтения	GetLastError
ERR_FILE_NOTTOWRITE	Файл должен быть открыт для записи	GetLastError
ERR_FILE_NOTTXT	Файл должен	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	быть открыт как текстовый	
ERR_FILE_NOTTXTORCSV	Файл должен быть открыт как текстовый или CSV	GetLastError
ERR_FILE_READERROR	Ошибка чтения файла	GetLastError
ERR_FILE_WRITEERROR	Не удалось записать ресурс в файл	GetLastError
ERR_FLOAT_ARRAY_ONLY	Должен быть массив типа float	GetLastError
ERR_FTP_SEND_FAILED	Не удалось отправить файл по ftp	GetLastError
ERR_FUNCTION_NOT_ALLOWED	Системная функция не разрешена для вызова	GetLastError
ERR_GLOBALVARIABLE_EXISTS	Глобальная переменная клиентс	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	кого терминала с таким именем уже существует	
ERR_GLOBALVARIABLE_NOT_FOUND	Глобальная переменная клиентского терминала не найдена	GetLastError
ERR_HISTORY_NOT_FOUND	Запрашиваемая история не найдена	GetLastError
ERR_HISTORY_WRONG_PROPERTY	Ошибочный идентификатор свойства истории	GetLastError
ERR_INCOMPATIBLE_ARRAYS	Копирование несовместимых массивов. Строковый массив может быть скопирован только в строковый, а	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	числовой массив - в числовой	
ERR_INCOMPATIBLE_FILE	Для строковых массивов должен быть текстовый файл, для остальных - бинарный	GetLastError
ERR_INDICATOR_CANNOT_ADD	Ошибка при добавлении индикатора	GetLastError
ERR_INDICATOR_CANNOT_APPLY	Индикатор не может быть применен к другому индикатору	GetLastError
ERR_INDICATOR_CANNOT_CREATE	Индикатор не может быть создан	GetLastError
ERR_INDICATOR_CUSTOM_NAME	Первым параметром в массиве	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	должно быть имя пользовательского индикатора	
ERR_INDICATOR_DATA_NOT_FOUND	Запрошенные данные не найдены	GetLastError
ERR_INDICATOR_NO_MEMORY	Недостаточно памяти для добавления индикатора	GetLastError
ERR_INDICATOR_PARAMETER_TYPE	Неправильный тип параметра в массиве при создании индикатора	GetLastError
ERR_INDICATOR_PARAMETERS_MISSING	Отсутствуют параметры при создании индикатора	GetLastError
ERR_INDICATOR_UNKNOWN_SYMBOL	Неизвестный символ	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_INDICATOR_WRONG_HANDLE	Ошибочный хэндл индикатора	GetLastError
ERR_INDICATOR_WRONG_INDEX	Ошибочный индекс запрашиваемого индикаторного буфера	GetLastError
ERR_INDICATOR_WRONG_PARAMETERS	Неправильное количество параметров при создании индикатора	GetLastError
ERR_INT_ARRAY_ONLY	Должен быть массив типа int	GetLastError
ERR_INTERNAL_ERROR	Неожиданная внутренняя ошибка	GetLastError
ERR_INVALID_ARRAY	Массив неподходящего типа, неподходящего размера или испорченный объект динамич	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	еского массива	
ERR_INVALID_DATETIME	Неправильное значение даты и/или времени	GetLastError
ERR_INVALID_FILEHANDLE	Файл с таким хэндлом уже был закрыт, либо не открывался вообще	GetLastError
ERR_INVALID_PARAMETER	Ошибочный параметр при вызове системной функции	GetLastError
ERR_INVALID_POINTER	Ошибочный указатель	GetLastError
ERR_INVALID_POINTER_TYPE	Ошибочный тип указателя	GetLastError
ERR_LONG_ARRAY_ONLY	Должен быть массив типа long	GetLastError
ERR_MAIL_SEND_FAILED	Не удалось отправить письмо	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_MARKET_LASTTIME_UNKNOWN	Время последнего тика неизвестно (тиков не было)	GetLastError
ERR_MARKET_NOT_SELECTED	Символ не выбран в Market Watch	GetLastError
ERR_MARKET_SELECT_ERROR	Ошибка добавления или удаления символа в Market Watch	GetLastError
ERR_MARKET_UNKNOWN_SYMBOL	Неизвестный символ	GetLastError
ERR_MARKET_WRONG_PROPERTY	Ошибочный идентификатор свойства символа	GetLastError
ERR_MQL5_WRONG_PROPERTY	Ошибочный идентификатор свойства программы	GetLastError
ERR_NO_STRING_DATE	В строке нет даты	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_NOT_ENOUGH_MEMORY	Недостаточно памяти для выполнения системной функции	GetLastError
ERR_NOTIFICATION_SEND_FAILED	Не удалось отправить уведомление	GetLastError
ERR_NOTIFICATION_TOO_FREQUENT	Слишком частая отправка уведомлений	GetLastError
ERR_NOTIFICATION_WRONG_PARAMETER	Неверный параметр для отправки уведомления - в функцию SendNotification() передали пустую строку или NULL	GetLastError
ERR_NOTIFICATION_WRONG_SETTINGS	Неверные настройки	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	уведомлений в терминале (не указан ID или не выставлено разрешение)	
ERR_NOTINITIALIZED_STRING	Неинициализированная строка	GetLastError
ERR_NUMBER_ARRAYS_ONLY	Должен быть числовой массив	GetLastError
ERR_OBJECT_ERROR	Ошибка при работе с графическим объектом	GetLastError
ERR_OBJECT_GETDATE_FAILED	Невозможно получить дату, соответствующую значению	GetLastError
ERR_OBJECT_GETVALUE_FAILED	Невозможно получить значение, соответствующее дате	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_OBJECT_NOT_FOUND	Графический объект не найден	GetLastError
ERR_OBJECT_WRONG_PROPERTY	Ошибочный идентификатор свойства графического объекта	GetLastError
ERR_ONEDIM_ARRAYS_ONLY	Должен быть одномерный массив	GetLastError
ERR_OPENCL_BUFFER_CREATE	Ошибка создания буфера OpenCL	GetLastError
ERR_OPENCL_CONTEXT_CREATE	Ошибка при создании контекста OpenCL	GetLastError
ERR_OPENCL_EXECUTE	Ошибка выполнения программы OpenCL	GetLastError
ERR_OPENCL_INTERNAL	Внутренняя ошибка при выполнении OpenCL	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_OPENCL_INVALID_HANDLE	Неправильный хэндл OpenCL	GetLastError
ERR_OPENCL_KERNEL_CREATE	Ошибка создания кернел - точки входа OpenCL	GetLastError
ERR_OPENCL_NOT_SUPPORTED	Функции OpenCL на данном компьютере не поддерживаются	GetLastError
ERR_OPENCL_PROGRAM_CREATE	Ошибка при компиляции программы OpenCL	GetLastError
ERR_OPENCL_QUEUE_CREATE	Ошибка создания очереди выполнения в OpenCL	GetLastError
ERR_OPENCL_SET_KERNEL_PARAMETER	Ошибка при установке параметров для кернел OpenCL (точки входа в програм	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	му OpenCL)	
ERR_OPENCL_TOO_LONG_KERNEL_NAME	Слишком длинное имя точки входа (кернел OpenCL)	GetLastError
ERR_OPENCL_WRONG_BUFFER_OFFSET	Неверное смещение в буфере OpenCL	GetLastError
ERR_OPENCL_WRONG_BUFFER_SIZE	Неверный размер буфера OpenCL	GetLastError
ERR_PLAY_SOUND_FAILED	Не удалось воспроизвести звук	GetLastError
ERR_RESOURCE_NAME_DUPLICATED	Совпадении имени динамического и статического ресурсов	GetLastError
ERR_RESOURCE_NAME_IS_TOO_LONG	Имя ресурса превышает 63 символа	GetLastError
ERR_RESOURCE_NOT_FOUND	Ресурс с таким именем	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	в EX5 не найден	
ERR_RESOURCE_UNSUPPORTED_TYPE	Неподдерживаемый тип ресурса или размер более 16 MB	GetLastError
ERR_SERIES_ARRAY	Таймсерия не может быть использована	GetLastError
ERR_SHORT_ARRAY_ONLY	Должен быть массив типа short	GetLastError
ERR_SMALL_ARRAY	Слишком маленький массив, стартовая позиция за пределами массива	GetLastError
ERR_SMALL_ASSERIES_ARRAY	Приемный массив объявлен как AS_SERIES, и он недостаточного размера	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_STRING_OUT_OF_MEMORY	Недостаточно памяти для строки	GetLastError
ERR_STRING_RESIZE_ERROR	Недостаточно памяти для перераспределения строки	GetLastError
ERR_STRING_SMALL_LEN	Длина строки меньше, чем ожидалось	GetLastError
ERR_STRING_TIME_ERROR	Ошибка преобразования строки в дату	GetLastError
ERR_STRING_TOO_BIGNUMBER	Слишком большое число, больше, чем ULONG_MAX	GetLastError
ERR_STRING_UNKNOWNTYPE	Неизвестный тип данных при конвертации в строку	GetLastError
ERR_STRING_ZEROADDED	К концу строки добавлен 0,	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	бесполезная операция	
ERR_STRINGPOS_OUTOFRANGE	Позиция за пределами строки	GetLastError
ERR_STRUCT_WITHOBJECTS_ORCLASS	Структура содержит объекты строк и/или динамических массивов и/или структуры с такими объектами и/или классы	GetLastError
ERR_SUCCESS	Операция выполнена успешно	GetLastError
ERR_TERMINAL_WRONG_PROPERTY	Ошибочный идентификатор свойства терминала	GetLastError
ERR_TOO_LONG_FILENAME	Слишком длинное имя файла	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_TOO_MANY_FILES	Не может быть открыто одновременно более 64 файлов	GetLastError
ERR_TOO_MANY_FORMATTERS	Форматных спецификаторов больше, чем параметров	GetLastError
ERR_TOO_MANY_PARAMETERS	Параметров больше, чем форматных спецификаторов	GetLastError
ERR_TRADE_DEAL_NOT_FOUND	Сделка не найдена	GetLastError
ERR_TRADE_DISABLED	Торговля для эксперта запрещена	GetLastError
ERR_TRADE_ORDER_NOT_FOUND	Ордер не найден	GetLastError
ERR_TRADE_POSITION_NOT_FOUND	Позиция не найдена	GetLastError
ERR_TRADE_SEND_FAILED	Не удалось отправи	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ть торговый запрос	
ERR_TRADE_WRONG_PROPERTY	Ошибочный идентификатор свойства торговли	GetLastError
ERR_USER_ERROR_FIRST	С этого кода начинаются ошибки, задаваемые пользователем	GetLastError
ERR_WEBREQUEST_CONNECT_FAILED	Не удалось подключиться к указанному URL	GetLastError
ERR_WEBREQUEST_INVALID_ADDRESS	URL не прошел проверку	GetLastError
ERR_WEBREQUEST_REQUEST_FAILED	Ошибка в результате выполнения HTTP запроса	GetLastError
ERR_WEBREQUEST_TIMEOUT	Превышен таймаут получения данных	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ERR_WRONG_DIRECTORYNAME	Ошибочное имя директории	GetLastError
ERR_WRONG_FILEHANDLE	Ошибочный хэндл файла	GetLastError
ERR_WRONG_FILENAME	Недопустимое имя файла	GetLastError
ERR_WRONG_FORMATSTRING	Ошибочная форматная строка	GetLastError
ERR_WRONG_INTERNAL_PARAMETER	Ошибочный параметр при внутреннем вызове функции клиентского терминала	GetLastError
ERR_WRONG_STRING_DATE	В строке ошибочная дата	GetLastError
ERR_WRONG_STRING_OBJECT	Испорченный объект строки	GetLastError
ERR_WRONG_STRING_PARAMETER	Испорченный параметр типа string	GetLastError
ERR_WRONG_STRING_TIME	В строке ошибочн	GetLastError

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ое время	
ERR_ZEROSIZE_ARRAY	Массив нулевой длины	GetLastError
FILE_ACCESS_DATE	Дата последнего доступа к файлу	FileGetInteger
FILE_ANSI	Строки типа ANSI (однобайтовые символы). Флаг используется при открытии файлов (FileOpen())	FileOpen
FILE_BIN	Двоичный режим чтения-записи (без преобразования из строки и в строку). Флаг используется при открытии файлов	FileOpen

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	(FileOpen())	
FILE_COMMON	Расположение файла в общей папке всех клиентских терминалов \Terminal\Common\Files. Флаг используется при открытии файлов (FileOpen()), копировании файлов (FileCopy(), FileMove()) и проверке существования файлов (FileIsExist())	FileOpen, FileCopy, FileMove, FileIsExist
FILE_CREATE_DATE	Дата создания	FileGetInteger
FILE_CSV	Файл типа csv (все записанные	FileOpen

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	элементы преобразуются к строкам соответствующего типа, unicode или ansi, и разделяются разделителем). Флаг используется при открытии файлов (FileOpen())	
FILE_END	Получение признака конца файла	FileGetInteger
FILE_EXISTS	Проверка на существование	FileGetInteger
FILE_IS_ANSI	Файл открыт как ANSI (смотри FILE_ANSI)	FileGetInteger
FILE_IS_BINARY	Файл открыт как бинарный (смотри	FileGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	FILE_BIN)	
FILE_IS_COMMON	Файл открыт в общей папке всех клиентских терминалов (смотри FILE_COMMON)	FileGetInteger
FILE_IS_CSV	Файл открыт как CSV (смотри FILE_CSV)	FileGetInteger
FILE_IS_READABLE	Файл открыт с возможностью чтения (смотри FILE_READ)	FileGetInteger
FILE_IS_TEXT	Файл открыт как текстовый (смотри FILE_TXT)	FileGetInteger
FILE_IS_WRITABLE	Файл открыт с возможностью записи (смотри FILE_WRITE)	FileGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
FILE_LINE_END	Получение признака конца строки	FileGetInteger
FILE_MODIFY_DATE	Дата последнего изменения	FileGetInteger
FILE_POSITION	Позиция указателя в файле	FileGetInteger
FILE_READ	Файл открывается для чтения. Флаг используется при открытии файлов (FileOpen()). При открытии файла обязательно должен быть указан флаг FILE_WRITE и/или флаг FILE_READ	FileOpen
FILE_REWRITE	Возможность перезап	FileCopy, FileMove

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	исывания файла функциями FileCopy() и FileMove(). Файл должен существовать или открываться для записи. В противном случае файл открыт не будет	
FILE_SHARE_READ	Совместный доступ по чтению со стороны нескольких программ. Флаг используется при открытии файлов (FileOpen()), но не заменяет при открытии файла необход	FileOpen

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	имости указать FILE_WRITE и/или флаг FILE_READ	
FILE_SHARE_WRITE	Совместный доступ по записи со стороны нескольких программ. Флаг используется при открытии файлов (FileOpen()), но не заменяет при открытии файла необходимости указать FILE_WRITE и/или флаг FILE_READ	FileOpen
FILE_SIZE	Размер файла в байтах	FileGetInteger
FILE_TXT	Простой текстов	FileOpen

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ый файл (тот же csv, однако разделитель не принимается во внимание). Флаг используется при открытии файлов (FileOpen())	
FILE_UNICODE	Строки типа UNICODE (двухбайтовые символы). Флаг используется при открытии файлов (FileOpen())	FileOpen
FILE_WRITE	Файл открывается для записи. Флаг используется при открытии файлов (FileOpen()).	FileOpen

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	При открытии файла обязательно должен быть указан флаг FILE_WRITE и/или флаг FILE_READ	
FLT_DIG	Количество значимых десятичных знаков	Константы числовых типов
FLT_EPSILON	Наименьшее число для которого выполняется условие 1.0+FLT_EPSILON != 1.0	Константы числовых типов
FLT_MANT_DIG	Количество битов в мантиссе	Константы числовых типов
FLT_MAX	Максимальное значение, которое может	Константы числовых типов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	быть представлено типом float	
FLT_MAX_10_EXP	Максимальное десятичное значение степени экспоненты	Константы числовых типов
FLT_MAX_EXP	Максимальное двоичное значение степени экспоненты	Константы числовых типов
FLT_MIN	Минимальное положительное значение, которое может быть представлено типом float	Константы числовых типов
FLT_MIN_10_EXP	Минимальное десятичное значение степени экспоненты	Константы числовых типов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
FLT_MIN_EXP	Минимальное двоичное значение степени экспоненты	Константы числовых типов
FRIDAY	Пятница	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
GANN_DOWN_TREND	Линия соответствует нисходящему тренду	ObjectSetInteger, ObjectGetInteger
GANN_UP_TREND	Линия соответствует восходящему тренду	ObjectSetInteger, ObjectGetInteger
GATORJAW_LINE	Линия челюстей	Линии индикаторов
GATORLIPS_LINE	Линия губ	Линии индикаторов
GATORTEETH_LINE	Линия зубов	Линии индикаторов
IDABORT	Выбрана кнопка Прервать (Abort)	MessageBox
IDCANCEL	Выбрана кнопка Отмена (Cancel)	MessageBox
IDCONTINUE	Выбрана кнопка	MessageBox

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	Продолжить (Continue)	
IDIGNORE	Выбрана кнопка Пропустить (Ignore)	MessageBox
IDNO	Выбрана кнопка Нет (No)	MessageBox
IDOK	Выбрана кнопка OK	MessageBox
IDRETRY	Выбрана кнопка Повтор (Retry)	MessageBox
IDTRYAGAIN	Выбрана кнопка Повторить (Try Again)	MessageBox
IDYES	Выбрана кнопка Да (Yes)	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	IndicatorCreate, IndicatorParameters

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	Wilder	
IND_ALLIGATOR	Alligator	IndicatorCreate, IndicatorParameters
IND_AMA	Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_AO	Awesome Oscillator	IndicatorCreate, IndicatorParameters
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

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
IND_FORCE	Force Index	IndicatorCreate, IndicatorParameters
IND_FRACTALS	Fractals	IndicatorCreate, IndicatorParameters
IND_FRAMA	Fractal Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_GATOR	Gator Oscillator	IndicatorCreate, IndicatorParameters
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	IndicatorCreate, IndicatorParameters

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	Oscillator	
IND_TEMA	Triple Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_TRIX	Triple Exponential Moving Averages Oscillator	IndicatorCreate, IndicatorParameters
IND_VIDYA	Variable Index Dynamic Average	IndicatorCreate, IndicatorParameters
IND_VOLUMES	Volumes	IndicatorCreate, IndicatorParameters
IND_WPR	Williams' Percent Range	IndicatorCreate, IndicatorParameters
INDICATOR_CALCULATIONS	Вспомогательные буферы для промежуточных вычислений	SetIndexBuffer
INDICATOR_COLOR_INDEX	Цвета отрисовки	SetIndexBuffer
INDICATOR_DATA	Данные для отрисовки	SetIndexBuffer
INDICATOR_DIGITS	Точность отображ	IndicatorSetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ения значений индикатора	
INDICATOR_HEIGHT	Фиксированная высота собственного окна индикатора (команда препроцессора #property indicator_height)	IndicatorSetInteger
INDICATOR_LEVELCOLOR	Цвет линии уровня	IndicatorSetInteger
INDICATOR_LEVELS	Количество уровней на окне индикатора	IndicatorSetInteger
INDICATOR_LEVELSTYLE	Стиль линии уровня	IndicatorSetInteger
INDICATOR_LEVELTEXT	Описание уровня	IndicatorSetString
INDICATOR_LEVELVALUE	Значение уровня	IndicatorSetDouble
INDICATOR_LEVELWIDTH	Толщина линии уровня	IndicatorSetInteger
INDICATOR_MAXIMUM	Максимум окна	IndicatorSetDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	индикатора	
INDICATOR_MINIMUM	Минимум окна индикатора	IndicatorSetDouble
INDICATOR_SHORTNAME	Короткое наименование индикатора	IndicatorSetString
INT_MAX	Максимальное значение, которое может быть представлено типом int	Константы числовых типов
INT_MIN	Минимальное значение, которое может быть представлено типом int	Константы числовых типов
INVALID_HANDLE	Некорректный хэндл	Прочие константы
IS_DEBUG_MODE	Признак работы mq5-программы в режиме отладки	Прочие константы

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
IS_PROFILE_MODE	Признак работы mq5-программы в режиме профилирования	Прочие константы
KIJUNSEN_LINE	Линия Kijun-sen	Линии индикаторов
LICENSE_DEMO	Демо-версия платного продукта из Маркета. Работает только в тестере стратегий	MQLInfoInteger
LICENSE_FREE	Бесплатная неограниченная версия	MQLInfoInteger
LICENSE_FULL	Купленная лицензионная версия допускает не менее 5 активаций. Продавец может увеличить разреше	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	нное число активаций	
LICENSE_TIME	Версия с ограниченной по времени лицензией	MQLInfoInteger
LONG_MAX	Максимальное значение, которое может быть представлено типом long	Константы числовых типов
LONG_MIN	Минимальное значение, которое может быть представлено типом long	Константы числовых типов
LOWER_BAND	Нижняя граница	Линии индикаторов
LOWER_HISTOGRAM	Нижняя гистограмма	Линии индикаторов
LOWER_LINE	Нижняя линия	Линии индикаторов
M_1_PI	1/pi	Математические константы
M_2_PI	2/pi	Математические константы

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
M_2_SQRTPI	2/sqrt(pi)	Математические константы
M_E	e	Математические константы
M_LN10	ln(10)	Математические константы
M_LN2	ln(2)	Математические константы
M_LOG10E	log10(e)	Математические константы
M_LOG2E	log2(e)	Математические константы
M_PI	pi	Математические константы
M_PI_2	pi/2	Математические константы
M_PI_4	pi/4	Математические константы
M_SQRT1_2	1/sqrt(2)	Математические константы
M_SQRT2	sqrt(2)	Математические константы
MAIN_LINE	Основная линия	Линии индикаторов
MB_ABORTRETRYIGNORE	Окно сообщения содержит три кнопки: Abort, Retry и Ignore	MessageBox
MB_CANCELTRYCONTINUE	Окно сообщения содержит три кнопки: Cancel, Try Again, Continue	MessageBox
MB_DEFBUTTON1	Первая кнопка MB_DEFBUTTON1 -	MessageBox

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	кнопка выбрана по умолчанию, если MB_DEFBUTTON2, MB_DEFBUTTON3, или MB_DEFBUTTON4 не определены	
MB_DEFBUTTON2	Вторая кнопка - кнопка по умолчанию	MessageBox
MB_DEFBUTTON3	Третья кнопка - кнопка по умолчанию	MessageBox
MB_DEFBUTTON4	Четвертая кнопка - кнопка по умолчанию	MessageBox
MB_ICONEXCLAMATION, MB_ICONWARNING	Изображение восклицательного знака	MessageBox
MB_ICONINFORMATION, MB_ICONASTERISK	Изображение, состоящее из строчно	MessageBox

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	го знака і в круге	
MB_ICONQUESTION	Изображение вопросительного знака	MessageBox
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	Изображение знака STOP	MessageBox
MB_OK	Окно сообщения содержит одну кнопку: OK. По умолчанию	MessageBox
MB_OKCANCEL	Окно сообщения содержит две кнопки: OK и Cancel	MessageBox
MB_RETRYCANCEL	Окно сообщения содержит две кнопки: Retry и Cancel	MessageBox
MB_YESNO	Окно сообщения содержит две кнопки: Yes и No	MessageBox

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
MB_YESNOCANCEL	Окно сообщения содержит три кнопки: Yes, No и Cancel	MessageBox
MINUSDI_LINE	Линия -DI	Линии индикаторов
MODE_EMA	Экспоненциальное усреднение	Методы скользящих
MODE_LWMA	Линейно-взвешенное усреднение	Методы скользящих
MODE_SMA	Простое усреднение	Методы скользящих
MODE_SMMA	Сглаженное усреднение	Методы скользящих
MONDAY	Понедельник	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
MQL_DEBUG	Признак работы запущенной программы в режиме отладки	MQLInfoInteger
MQL_DLLS_ALLOWED	Разрешение на использование	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	DLL для данной запущенной программы	
MQL_FRAME_MODE	Признак работы запущенного эксперта на графике в режиме сбора фреймов результатов оптимизации	MQLInfoInteger
MQL_LICENSE_TYPE	Тип лицензии модуля EX5. Лицензия относится именно к тому модулю EX5, из которого делается запрос с помощью MQLInfoInteger(MQL_LIC	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ENSE_TYPE).	
MQL_MEMORY_LIMIT	Максимально возможный объём динамической памяти для MQL5-программы в MB	MQLInfoInteger
MQL_MEMORY_USED	Размер использованной памяти MQL5-программой в MB	MQLInfoInteger
MQL_OPTIMIZATION	Признак работы запущенной программы в процессе оптимизации	MQLInfoInteger
MQL_PROFILER	Признак работы запущенной программы в режиме профилирования кода	MQLInfoInteger
MQL_PROGRAM_NAME	Имя запущенной	MQLInfoString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	MQL5-программы	
MQL_PROGRAM_PATH	Путь для данной запущенной программы	MQLInfoString
MQL_PROGRAM_TYPE	Тип mql5-программы	MQLInfoInteger
MQL_SIGNALS_ALLOWED	Разрешение на работу с сигналами данной запущенной программы	MQLInfoInteger
MQL_TESTER	Признак работы запущенной программы в тестере	MQLInfoInteger
MQL_TRADE_ALLOWED	Разрешение на торговлю для данной запущенной программы	MQLInfoInteger
MQL_VISUAL_MODE	Признак работы запущенной	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	программы в визуальном режиме тестирования	
NULL	Ноль любого типа	Прочие константы
OBJ_ALL_PERIODS	Объект рисуется на всех таймфреймах	ObjectSetInteger, ObjectGetInteger
OBJ_ARROW	Объект "Стрелка"	Типы объектов
OBJ_ARROW_BUY	Знак "Buy"	Типы объектов
OBJ_ARROW_CHECK	Знак "Птичка" (галка)	Типы объектов
OBJ_ARROW_DOWN	Знак "Стрелка вниз"	Типы объектов
OBJ_ARROW_LEFT_PRICE	Левая ценовая метка	Типы объектов
OBJ_ARROW_RIGHT_PRICE	Правая ценовая метка	Типы объектов
OBJ_ARROW_SELL	Знак "Sell"	Типы объектов
OBJ_ARROW_STOP	Знак "Стоп"	Типы объектов
OBJ_ARROW_THUMB_DOWN	Знак "Плохо" (большой палец вниз)	Типы объектов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
OBJ_ARROW_THUMB_UP	Знак "Хорошо" (большой палец вверх)	Типы объектов
OBJ_ARROW_UP	Знак "Стрелка вверх"	Типы объектов
OBJ_ARROWED_LINE	Объект "Линия со стрелкой"	Типы объектов
OBJ_BITMAP	Объект "Рисунок"	Типы объектов
OBJ_BITMAP_LABEL	Объект "Графическая метка"	Типы объектов
OBJ_BUTTON	Объект "Кнопка"	Типы объектов
OBJ_CHANNEL	Равноудаленный канал	Типы объектов
OBJ_CHART	Объект "График"	Типы объектов
OBJ_CYCLES	Циклические линии	Типы объектов
OBJ_EDIT	Объект "Поле ввода"	Типы объектов
OBJ_ELLIOTWAVE3	3-волновка Эллиота	Типы объектов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
OBJ_ELLIOTWAVE5	5-волновка Эллиота	Типы объектов
OBJ_ELLIPSE	Эллипс	Типы объектов
OBJ_EVENT	Объект "Событие", соответствующий событию в экономическом календаре	Типы объектов
OBJ_EXPANSION	Расширение Фибоначчи	Типы объектов
OBJ_FIBO	Уровни Фибоначчи	Типы объектов
OBJ_FIBOARC	Дуги Фибоначчи	Типы объектов
OBJ_FIBOCHANNEL	Канал Фибоначчи	Типы объектов
OBJ_FIBOFAN	Веер Фибоначчи	Типы объектов
OBJ_FIBOTIMES	Временные зоны Фибоначчи	Типы объектов
OBJ_GANNFAN	Веер Ганна	Типы объектов
OBJ_GANNGRID	Сетка Ганна	Типы объектов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
OBJ_GANNLINE	Линия Ганна	Типы объектов
OBJ_HLINE	Горизонтальная линия	Типы объектов
OBJ_LABEL	Объект "Текстовая метка"	Типы объектов
OBJ_NO_PERIODS	Объект не показывается ни на одном таймфрейме	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_D1	Объект рисуется на дневных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H1	Объект рисуется на 1-часовых графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H12	Объект рисуется на 12-часовых графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H2	Объект рисуется на 2-часовых графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H3	Объект рисуется	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	на 3-часовых графиках	
OBJ_PERIOD_H4	Объект рисуется на 4-часовых графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H6	Объект рисуется на 6-часовых графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H8	Объект рисуется на 8-часовых графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M1	Объект рисуется на 1-минутных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M10	Объект рисуется на 10-минутных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M12	Объект рисуется на 12-минутных графиках	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
OBJ_PERIOD_M15	Объект рисуется на 15-минутных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M2	Объект рисуется на 2-минутных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M20	Объект рисуется на 20-минутных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M3	Объект рисуется на 3-минутных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M30	Объект рисуется на 30-минутных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M4	Объект рисуется на 4-минутных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M5	Объект рисуется	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	на 5-минутных графиках	
OBJ_PERIOD_M6	Объект рисуется на 6-минутных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_MN1	Объект рисуется на месячных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_W1	Объект рисуется на недельных графиках	ObjectSetInteger, ObjectGetInteger
OBJ_PITCHFORK	Вилы Эндрюса	Типы объектов
OBJ_RECTANGLE	Прямоугольник	Типы объектов
OBJ_RECTANGLE_LABEL	Объект "Прямоугольная метка"для создания и оформления пользовательского графического	Типы объектов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	интерфейса.	
OBJ_REGRESSION	Канал на линейной регрессии	Типы объектов
OBJ_STDDEVCHANNEL	Канал стандартного отклонения	Типы объектов
OBJ_TEXT	Объект "Текст"	Типы объектов
OBJ_TREND	Трендовая линия	Типы объектов
OBJ_TRENDBYANGLE	Трендовая линия по углу	Типы объектов
OBJ_TRIANGLE	Треугольник	Типы объектов
OBJ_VLINE	Вертикальная линия	Типы объектов
OBJPROP_ALIGN	Горизонтальное выравнивание текста в объекте "Поле ввода" (OBJ_EDIT)	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANCHOR	Положение точки привязки	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	графического объекта	
OBJPROP_ANGLE	Угол. Для объектов с еще не заданным углом, созданных из программы, значение равно EMPTY_VALUE	ObjectSetDouble, ObjectGetDouble
OBJPROP_ARROWCODE	Код стрелки для объекта "Стрелка"	ObjectSetInteger, ObjectGetInteger
OBJPROP_BACK	Объект на заднем плане	ObjectSetInteger, ObjectGetInteger
OBJPROP_BGCOLOR	Цвет фона для OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	ObjectSetInteger, ObjectGetInteger
OBJPROP_BMPFILE	Имя BMP-файла для объекта "Графич	ObjectSetString, ObjectGetString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ческая метка". Смотри также Ресурсы	
OBJPROP_BORDER_COLOR	Цвет рамки для объекта OBJ_EDIT и OBJ_BUTTON	ObjectSetInteger, ObjectGetInteger
OBJPROP_BORDER_TYPE	Тип рамки для объекта "Прямоугольная рамка"	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_ID	Идентификатор объекта "График" (OBJ_CHART). Позволяет работать со свойствами этого объекта как с обычным графиком с помощью функций из раздела Операции с	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	графиками, но есть некоторые исключения.	
OBJPROP_CHART_SCALE	Масштаб для объекта "График"	ObjectSetInteger, ObjectGetInteger
OBJPROP_COLOR	Цвет	ObjectSetInteger, ObjectGetInteger
OBJPROP_CORNER	Угол графика для привязки графического объекта	ObjectSetInteger, ObjectGetInteger
OBJPROP_CREATETIME	Время создания объекта	ObjectSetInteger, ObjectGetInteger
OBJPROP_DATE_SCALE	Признак отображения шкалы времени для объекта "График"	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEGREE	Уровень волновой разметки Эллиота	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEVIATION	Отклонение для канала	ObjectSetDouble, ObjectGetDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	стандартного отклонения	
OBJPROP_DIRECTION	Тренд объекта Ганна	ObjectSetInteger, ObjectGetInteger
OBJPROP_DRAWLINES	Отображение линий для волновой разметки Эллиота	ObjectSetInteger, ObjectGetInteger
OBJPROP_ELLIPSE	Отображение полного эллипса для объекта "Дуги Фибоначчи" (OBJ_FIBOARC)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FILL	Заливка объекта цветом (для OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
OBJPROP_FONT	Шрифт	ObjectSetString, ObjectGetString
OBJPROP_FONTSIZE	Размер шрифта	ObjectSetInteger, ObjectGetInteger
OBJPROP_HIDDEN	Запрет на показ имени графического объекта в списке объектов из меню терминала "Графики" - "Объекты" - "Список объектов". Значение true позволяет скрыть ненужный для пользователя объект из списка. По умолчанию true устанавливается для объектов, которые отображают события	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	календаря, историю торговли, а также для созданных из MQL5-программы. Для того чтобы увидеть такие графические объекты и получить доступ к их свойствам, нужно нажать кнопку "Все" в окне "Список объектов".	
OBJPROP_LEVELCOLOR	Цвет линии-уровня	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELS	Количество уровней	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELSTYLE	Стиль линии-уровня	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELTEXT	Описание уровня	ObjectSetString, ObjectGetString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
OBJPROP_LEVELVALUE	Значение уровня	ObjectSetDouble, ObjectGetDouble
OBJPROP_LEVELWIDTH	Толщина линии-уровня	ObjectSetInteger, ObjectGetInteger
OBJPROP_NAME	Имя объекта	ObjectSetString, ObjectGetString
OBJPROP_PERIOD	Период для объекта "График"	ObjectSetInteger, ObjectGetInteger
OBJPROP_PRICE	Координата цены	ObjectSetDouble, ObjectGetDouble
OBJPROP_PRICE_SCALE	Признак отображения ценовой шкалы для объекта "График"	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY	Вертикальная линия продолжается на все окна графика	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_LEFT	Луч продолжается влево	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_RIGHT	Луч продолжается вправо	ObjectSetInteger, ObjectGetInteger
OBJPROP_READONLY	Возможность	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	редактирования текста в объекте Edit	
OBJPROP_SCALE	Масштаб (свойство объектов Ганна и объекта "Дуги Фибоначчи")	ObjectSetDouble, ObjectGetDouble
OBJPROP_SELECTABLE	Доступность объекта	ObjectSetInteger, ObjectGetInteger
OBJPROP_SELECTED	Выделенность объекта	ObjectSetInteger, ObjectGetInteger
OBJPROP_STATE	Состояние кнопки (Нажата/Отжата)	ObjectSetInteger, ObjectGetInteger
OBJPROP_STYLE	Стиль	ObjectSetInteger, ObjectGetInteger
OBJPROP_SYMBOL	Символ для объекта "График"	ObjectSetString, ObjectGetString
OBJPROP_TEXT	Описание объекта (текст, содержащийся в объекте)	ObjectSetString, ObjectGetString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
OBJPROP_TIME	Координата времени	ObjectSetInteger, ObjectGetInteger
OBJPROP_TIMEFRAMES	Видимость объекта на таймфреймах	ObjectSetInteger, ObjectGetInteger
OBJPROP_TOOLTIP	Текст всплывающей подсказки. Если свойство не задано, то показывается подсказка, автоматически сформированная терминалом. Можно отключить показ подсказки, установив для нее значение "\n" (перевод строки)	ObjectSetString, ObjectGetString
OBJPROP_TYPE	Тип объекта	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
OBJPROP_WIDTH	Толщина линии	ObjectSetInteger, ObjectGetInteger
OBJPROP_XDISTANCE	Дистанция в пикселях по оси X от угла привязки (см. примечание)	ObjectSetInteger, ObjectGetInteger
OBJPROP_XOFFSET	X-координата левого верхнего угла прямоугольной области видимости в графических объектах "Графическая метка" и "Рисунок" (OBJ_BITMAP_LABEL и OBJ_BITMAP). Значение задается в пикселях относительного верхнего левого	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	угла исходного изображения.	
OBJPROP_XSIZE	Ширина объекта по оси X в пикселях. Задается для объектов OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	ObjectSetInteger, ObjectGetInteger
OBJPROP_YDISTANCE	Дистанция в пикселях по оси Y от угла привязки (см. примечание)	ObjectSetInteger, ObjectGetInteger
OBJPROP_YOFFSET	Y-координата левого	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	верхнего угла прямоугольной области видимости в графических объектах "Графическая метка" и "Рисунок" (OBJ_BITMAP_LABEL и OBJ_BITMAP). Значение задается в пикселях относительного верхнего левого угла исходного изображения.	
OBJPROP_YSIZE	Высота объекта по оси Y в пикселях. Задается для объектов OBJ_LABEL (read	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_ZORDER	Приоритет графического объекта на получение события нажатия мышки на графике (CHARTEVENT_CLICK). По умолчанию при создании значение выставляется равным нулю, но при необходимости можно повысит	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ь приоритет. При наложении объектов друг на друга событие CHARTEVENT_CLICK получит только один объект, чей приоритет выше остальных.	
ORDER_COMMENT	Комментарий	OrderGetString, HistoryOrderGetString
ORDER_FILLING_FOK	Данная политика исполнения означает, что ордер может быть исполнен исключительно в указанном объеме. Если на рынке в данный момент не присутст	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	вует достаточного объема финансового инструмента, то ордер не будет исполнен. Необходимый объем может быть составлен из нескольких предложений, доступных в данный момент на рынке.	
ORDER_FILLING_IOC	Означает согласие совершить сделку по максимально доступному на рынке объему в пределах указанн	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ого в ордере. В случае невозможности полного исполнения ордер будет исполнен на доступный объем, а неисполненный объем ордера будет отменен.	
ORDER_FILLING_RETURN	Данный режим используется для рыночных (ORDER_TYPE_BUY и ORDER_TYPE_SELL), лимитных и стоп-лимитных ордеров (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SE	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	LL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT и ORDER_TYPE_SELL_STOP_LIMIT) и только в режимах "Исполнение по рынку" и "Биржевое исполнение". В случае частичного исполнения рыночный или лимитный ордер с остаточным объемом не снимается, а продолжает действовать. Для ордеров ORDER_TYPE_BUY_STOP_LIMIT и ORDER_	

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	TYPE_SELL_STOP_LIMIT при активации будет создан соответствующий лимитный ордер ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT с типом исполнения ORDER_FILLING_RETURN.	
ORDER_MAGIC	Идентификатор эксперта выставившего ордер (предназначен для того, чтобы каждый эксперт выставлял свой собственный уникальный номер)	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ORDER_POSITION_ID	Идентификатор позиции, который ставится на ордере при его исполнении. Каждый исполненный ордер порождает сделку, которая открывает новую или изменяет уже существующую позицию. Идентификатор этой позиции и устанавливается исполненному ордеру в этот момент.	OrderGetInteger, HistoryOrderGetInteger
ORDER_PRICE_CURRENT	Текущая цена по символу ордера	OrderGetDouble, HistoryOrderGetDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ORDER_PRICE_OPEN	Цена, указанная в ордере	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_STOPLIMIT	Цена постановки Limit ордера при срабатывании StopLimit ордера	OrderGetDouble, HistoryOrderGetDouble
ORDER_SL	Уровень Stop Loss	OrderGetDouble, HistoryOrderGetDouble
ORDER_STATE	Статус ордера	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_CANCELED	Ордер снят клиентом	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_EXPIRED	Ордер снят по истечении срока его действия	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_FILLED	Ордер выполнен полностью	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PARTIAL	Ордер выполнен частично	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PLACED	Ордер принят	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ORDER_STATE_REJECTED	Ордер отклонен	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_ADD	Ордер в состоянии регистрации (выставление в торговую систему)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_CANCEL	Ордер в состоянии удаления (удаление из торговой системы)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_MODIFY	Ордер в состоянии модификации (изменение его параметров)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_STARTED	Ордер проверен на корректность, но еще не принят брокером	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
ORDER_SYMBOL	Символ, по которому выставлен ордер	OrderGetString, HistoryOrderGetString
ORDER_TIME_DAY	Ордер будет действовать только в течение текущего торгового дня	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE	Время исполнения или снятия ордера	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE_MSC	Время исполнения/снятия ордера в миллисекундах с 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_EXPIRATION	Время истечения ордера	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_GTC	Ордер будет находится в очереди до тех пор, пока не	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	будет снят	
ORDER_TIME_SETUP	Время постановки ордера	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP_MSC	Время установки ордера на исполнение в миллисекундах с 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED	Ордер будет действовать до даты истечения	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED_DAY	Ордер будет действовать до 23:59:59 указанного дня. Если это время не попадает на торговую сессию, истечение наступит в ближайшее	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	торговое время.	
ORDER_TP	Уровень Take Profit	OrderGetDouble, HistoryOrderGetDouble
ORDER_TYPE	Тип ордера	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY	Рыночный ордер на покупку	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_LIMIT	Отложенный ордер Buy Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP	Отложенный ордер Buy Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP_LIMIT	По достижении цены ордера выставляется отложенный ордер Buy Limit по цене StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_FILLING	Тип исполнения по остатку	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL	Рыночный	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ордер на продажу	
ORDER_TYPE_SELL_LIMIT	Отложенный ордер Sell Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP	Отложенный ордер Sell Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP_LIMIT	По достижении цены ордера выставляется отложенный ордер Sell Limit по цене StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_TIME	Время жизни ордера	OrderGetInteger, HistoryOrderGetInteger
ORDER_VOLUME_CURRENT	Невыполненный объем	OrderGetDouble, HistoryOrderGetDouble
ORDER_VOLUME_INITIAL	Первоначальный объем при постановке ордера	OrderGetDouble, HistoryOrderGetDouble
PERIOD_CURRENT	Текущий период	Периоды графиков
PERIOD_D1	1 день	Периоды графиков

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
PERIOD_H1	1 час	Периоды графиков
PERIOD_H12	12 часов	Периоды графиков
PERIOD_H2	2 часа	Периоды графиков
PERIOD_H3	3 часа	Периоды графиков
PERIOD_H4	4 часа	Периоды графиков
PERIOD_H6	6 часов	Периоды графиков
PERIOD_H8	8 часов	Периоды графиков
PERIOD_M1	1 минута	Периоды графиков
PERIOD_M10	10 минут	Периоды графиков
PERIOD_M12	12 минут	Периоды графиков
PERIOD_M15	15 минут	Периоды графиков
PERIOD_M2	2 минуты	Периоды графиков
PERIOD_M20	20 минут	Периоды графиков
PERIOD_M3	3 минуты	Периоды графиков
PERIOD_M30	30 минут	Периоды графиков
PERIOD_M4	4 минуты	Периоды графиков
PERIOD_M5	5 минут	Периоды графиков
PERIOD_M6	6 минут	Периоды графиков
PERIOD_MN1	1 месяц	Периоды графиков
PERIOD_W1	1 неделя	Периоды графиков
PLOT_ARROW	Код стрелки для стиля	PlotIndexSetInteger, PlotIndexGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	DRAW_ARROW	
PLOT_ARROW_SHIFT	Смещение стрелок по вертикали для стиля DRAW_ARROW	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_COLOR_INDEXES	Количество цветов	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_BEGIN	Количество начальных баров без отрисовки и значений в DataWindow	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_TYPE	Тип графического построения	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_EMPTY_VALUE	Пустое значение для построения, для которого нет отрисовки	PlotIndexSetDouble
PLOT_LABEL	Имя индикаторной графиче	PlotIndexSetString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ской серии для отображения в окне DataWindow. Для сложных графических стилей, требующих для отображения несколько индикаторных буферов, имена для каждого буфера можно задать с использованием ";" в качестве разделителя. Пример кода приведен в DRAW_CANDLES	
PLOT_LINE_COLOR	Индекс буфера, содержащего цвет	PlotIndexSetInteger, PlotIndexGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	отрисовки	
PLOT_LINE_STYLE	Стиль линии отрисовки	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_WIDTH	Толщина линии отрисовки	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHIFT	Сдвиг графического построения индикатора по оси времени в барах	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHOW_DATA	Признак отображения значений построения в окне DataWindow	PlotIndexSetInteger, PlotIndexGetInteger
PLUSDI_LINE	Линия +DI	Линии индикаторов
POINTER_AUTOMATIC	Указатель любого объекта, созданного автоматически (без использ	CheckPointer

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ования new())	
POINTER_DYNAMIC	Указатель объекта, созданного оператором new	CheckPointer
POINTER_INVALID	Некорректный указатель	CheckPointer
POSITION_COMMENT	Комментарий к позиции	PositionGetString
POSITION_COMMISSION	Комиссия	PositionGetDouble
POSITION_IDENTIFIER	Идентификатор позиции - это уникальное число, которое присваивается каждой вновь открытой позиции и не изменяется в течение всей ее жизни. Переворот позиции не изменяе	PositionGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	т идентификатора позиции.	
POSITION_MAGIC	Magic number для позиции (смотри ORDER_MAGIC)	PositionGetInteger
POSITION_PRICE_CURRENT	Текущая цена по символу	PositionGetDouble
POSITION_PRICE_OPEN	Цена позиции	PositionGetDouble
POSITION_PROFIT	Текущая прибыль	PositionGetDouble
POSITION_SL	Уровень Stop Loss для открытой позиции	PositionGetDouble
POSITION_SWAP	Накопленный своп	PositionGetDouble
POSITION_SYMBOL	Символ, по которому открыта позиция	PositionGetString
POSITION_TIME	Время открытия позиции	PositionGetInteger
POSITION_TIME_MSC	Время открытия позиции	PositionGetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	в миллисекундах с 01.01.1970	
POSITION_TIME_UPDATE	Время изменения позиции в секундах с 01.01.1970	PositionGetInteger
POSITION_TIME_UPDATE_MSC	Время изменения позиции в миллисекундах с 01.01.1970	PositionGetInteger
POSITION_TP	Уровень Take Profit для открытой позиции	PositionGetDouble
POSITION_TYPE	Тип позиции	PositionGetInteger
POSITION_TYPE_BUY	Покупка	PositionGetInteger
POSITION_TYPE_SELL	Продажа	PositionGetInteger
POSITION_VOLUME	Объем позиции	PositionGetDouble
PRICE_CLOSE	Цена закрытия	Ценовые константы
PRICE_HIGH	Максимальная	Ценовые константы

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	за период цена	
PRICE_LOW	Минимальная за период цена	Ценовые константы
PRICE_MEDIAN	Медианная цена, (high+low)/2	Ценовые константы
PRICE_OPEN	Цена открытия	Ценовые константы
PRICE_TYPICAL	Типичная цена, (high+low+close)/3	Ценовые константы
PRICE_WEIGHTED	Среднезвешенная цена, (high+low+close+close)/4	Ценовые константы
PROGRAM_EXPERT	Эксперт	MQLInfoInteger
PROGRAM_INDICATOR	Индикатор	MQLInfoInteger
PROGRAM_SCRIPT	Скрипт	MQLInfoInteger
REASON_ACCOUNT	Активирован другой счет либо произошло переподключение к торговому	UninitializeReason, OnDeinit

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	серверу вследствие изменения настроек счета	
REASON_CHARTCHANGE	Символ или период графика был изменен	UninitializeReason, OnDeinit
REASON_CHARTCLOSE	График закрыт	UninitializeReason, OnDeinit
REASON_CLOSE	Терминал был закрыт	UninitializeReason, OnDeinit
REASON_INITFAILED	Признак того, что обработчик OnInit() вернул ненулевое значение	UninitializeReason, OnDeinit
REASON_PARAMETERS	Входные параметры были изменены пользователем	UninitializeReason, OnDeinit
REASON_PROGRAM	Эксперт прекратил свою работу, вызвав функцию	UninitializeReason, OnDeinit

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ExpertRemove()	
REASON_RECOMPILE	Программа перекомпилирована	UninitializeReason, OnDeinit
REASON_REMOVE	Программа удалена с графика	UninitializeReason, OnDeinit
REASON_TEMPLATE	Применен другой шаблон графика	UninitializeReason, OnDeinit
SATURDAY	Суббота	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SEEK_CUR	Текущая позиция файлового указателя	FileSeek
SEEK_END	Конец файла	FileSeek
SEEK_SET	Начало файла	FileSeek
SENKOUSPANA_LINE	Линия Senkou Span A	Линии индикаторов
SENKOUSPANB_LINE	Линия Senkou Span B	Линии индикаторов
SERIES_BARS_COUNT	Количество баров по символу -	SeriesInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	периоду на данный момент	
SERIES_FIRSTDATE	Самая первая дата по символу - периоду на данный момент	SeriesInfoInteger
SERIES_LASTBAR_DATE	Время открытия последнего бара по символу - периоду	SeriesInfoInteger
SERIES_SERVER_FIRSTDATE	Самая первая дата в истории по символу на сервере независимо от периода	SeriesInfoInteger
SERIES_SYNCHRONIZED	Признак синхронизированности данных по символу/периоду на данный момент	SeriesInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
SERIES_TERMINAL_FIRSTDATE	Самая первая дата в истории по символу в клиентском терминале независимо от периода	SeriesInfoInteger
SHORT_MAX	Максимальное значение, которое может быть представлено типом short	Константы числовых типов
SHORT_MIN	Минимальное значение, которое может быть представлено типом short	Константы числовых типов
SIGNAL_BASE_AUTHOR_LOGIN	Логин автора сигнала	SignalBaseGetString
SIGNAL_BASE_BALANCE	Баланс счета	SignalBaseGetDouble
SIGNAL_BASE_BROKER	Наименование брокера	SignalBaseGetString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	(компании)	
SIGNAL_BASE_BROKER_SERVER	Сервер брокера	SignalBaseGetString
SIGNAL_BASE_CURRENCY	Валюта счета сигнала	SignalBaseGetString
SIGNAL_BASE_DATE_PUBLISHED	Дата публикации сигнала (когда стал доступен для подписки)	SignalBaseGetInteger
SIGNAL_BASE_DATE_STARTED	Дата начала мониторинга сигнала	SignalBaseGetInteger
SIGNAL_BASE_EQUITY	Средства на счете	SignalBaseGetDouble
SIGNAL_BASE_GAIN	Прирост счета в процентах	SignalBaseGetDouble
SIGNAL_BASE_ID	ID сигнала	SignalBaseGetInteger
SIGNAL_BASE_LEVERAGE	Плечо торгового счета	SignalBaseGetInteger
SIGNAL_BASE_MAX_DRAWDOWN	Максимальная просадка	SignalBaseGetDouble
SIGNAL_BASE_NAME	Имя сигнала	SignalBaseGetString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
SIGNAL_BASE_PIPS	Результат торговли в пипсах	SignalBaseGetInteger
SIGNAL_BASE_PRICE	Цена подписки на сигнал	SignalBaseGetDouble
SIGNAL_BASE_RATING	Позиция в рейтинге сигналов	SignalBaseGetInteger
SIGNAL_BASE_ROI	Значение ROI (Return on Investment) сигнала в %	SignalBaseGetDouble
SIGNAL_BASE_SUBSCRIBERS	Количество подписчиков	SignalBaseGetInteger
SIGNAL_BASE_TRADE_MODE	Тип счета (0-реальный счет, 1-демо-счет, 2-конкурсный счет)	SignalBaseGetInteger
SIGNAL_BASE_TRADES	Количество трейдов	SignalBaseGetInteger
SIGNAL_INFO_CONFIRMATIONS_DISABLED	Флаг разрешения	SignalInfoGetInteger, SignalInfoSetInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	синхронизации без показа диалога подтверждения	
SIGNAL_INFO_COPY_SLTP	Флаг копирования Stop Loss и Take Profit	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_DEPOSIT_PERCENT	Ограничения по депозиту (в %)	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_EQUITY_LIMIT	Процент для конвертации объема сделки	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_ID	id сигнала, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_NAME	Имя сигнала, r/o	SignalInfoGetString
SIGNAL_INFO_SLIPPAGE	Величина проскальзывания, с которым выставляются рыночные ордера при синхронизации	SignalInfoGetDouble, SignalInfoSetDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	позиций и копировании сделок	
SIGNAL_INFO_SUBSCRIPTION_ENABLED	Флаг разрешения на копирование сделок по подписке	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_TERMS_AGREE	Флаг согласия с условиями использования сервиса "Сигналы", r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_VOLUME_PERCENT	Значение ограничения по средствам для сигнала, r/o	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_LINE	Сигнальная линия	Линии индикаторов
STAT_BALANCE_DD	Максимальная просадка баланса в деньгах. В процессе	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	торговли баланс может испытать множество просадок, берется наибольшее значение.	
STAT_BALANCE_DD_RELATIVE	Просадка баланса в деньгах, которая была зафиксирована в момент максимальной просадки баланса в процентах (STAT_BALANCE_DDREL_PERCENT).	TesterStatistics
STAT_BALANCE_DDREL_PERCENT	Максимальная просадка баланса в процент	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ах. В процессе торговли баланс может испытать множество просадок, для каждой фиксируется относительное значение просадки в процентах. Возвращается наибольшее значение	
STAT_BALANCEDD_PERCENT	Просадка баланса в процентах, которая была зафиксирована в момент максимальной просадки баланса в	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	деньгах (STAT_BALANCE_DD).	
STAT_BALANCEMIN	Минимальное значение баланса	TesterStatistics
STAT_CONLOSSMAX	Максимальный убыток в последовательности убыточных трейдов. Значение меньше или равно нулю	TesterStatistics
STAT_CONLOSSMAX_TRADES	Количество трейдов, сформировавших STAT_CONLOSSMAX (максимальный убыток в последовательности убыточных	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	трейдов)	
STAT_CONPROFITMAX	Максимальная прибыль в последовательности прибыльных трейдов. Значение больше или равно нулю	TesterStatistics
STAT_CONPROFITMAX_TRADES	Количество трейдов, сформировавших STAT_CONPROFITMAX (максимальная прибыль в последовательности прибыльных трейдов)	TesterStatistics
STAT_CUSTOM_ONTESTER	Значение рассчитанного пользовательск	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ого критерия оптимизации, возвращаемого функцией OnTester()	
STAT_DEALS	Количество совершенных сделок	TesterStatistics
STAT_EQUITY_DD	Максимальная просадка средств в деньгах. В процессе торговли средства могут испытать множество просадок, берется наибольшее значение.	TesterStatistics
STAT_EQUITY_DD_RELATIVE	Просадка средств в деньгах	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	, которая была зафиксирована в момент максимальной просадки средств в процентах (STAT_EQUITY_DDREL_PERCENT).	
STAT_EQUITY_DDREL_PERCENT	Максимальная просадка средств в процентах. В процессе торговли средства могут испытать множество просадок, для каждой фиксируется относительное значение просадк	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	и в процентах. Возвращается наибольшее значение	
STAT_EQUITYDD_PERCENT	Просадка средств в процентах, которая была зафиксирована в момент максимальной просадки средств в деньгах (STAT_EQUITY_DD).	TesterStatistics
STAT_EQUITYMIN	Минимальное значение собственных средств	TesterStatistics
STAT_EXPECTED_PAYOFF	Математическое ожидание выигрыша	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
STAT_GROSS_LOSS	Общий убыток, сумма всех убыточных (отрицательных) трейдов. Значение меньше или равно нулю	TesterStatistics
STAT_GROSS_PROFIT	Общая прибыль, сумма всех прибыльных (положительных) трейдов. Значение больше или равно нулю	TesterStatistics
STAT_INITIAL_DEPOSIT	Значение начального депозита	TesterStatistics
STAT_LONG_TRADES	Длинные трейды	TesterStatistics
STAT_LOSS_TRADES	Убыточные трейды	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
STAT_LOSSTRADES_AVGCON	Средняя длина убыточной серии трейдов	TesterStatistics
STAT_MAX_CONLOSS_TRADES	Количество трейдов в самой длинной серии убыточных трейдов STAT_MAX_CONLOSSES	TesterStatistics
STAT_MAX_CONLOSSES	Общий убыток в самой длинной серии убыточных трейдов	TesterStatistics
STAT_MAX_CONPROFIT_TRADES	Количество трейдов в самой длинной серии прибыльных трейдов STAT_MAX_CONWINS	TesterStatistics
STAT_MAX_CONWINS	Общая прибыль в самой длинной серии прибыльных трейдов	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
STAT_MAX_LOSSTRADE	Максимальный убыток - наименьшее значение среди всех убыточных трейдов. Значение меньше или равно нулю	TesterStatistics
STAT_MAX_PROFITTRADE	Максимальная прибыль - наибольшее значение среди всех прибыльных трейдов. Значение больше или равно нулю	TesterStatistics
STAT_MIN_MARGINLEVEL	Минимальное достигнутое значение уровня маржи	TesterStatistics
STAT_PROFIT	Чистая прибыль	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	по окончании тестирования, сумма STAT_GROSS_PROFIT и STAT_GROSS_LOSS (STAT_GROSS_LOSS всегда меньше или равно нулю)	
STAT_PROFIT_FACTOR	Прибыльность - отношение STAT_GROSS_PROFIT/STAT_GROSS_LOSS. Если STAT_GROSS_LOSS=0, то прибыльность принимает значение DBL_MAX	TesterStatistics
STAT_PROFIT_LONGTRADES	Длинные прибыль	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ные трейды	
STAT_PROFIT_SHORTTRADES	Короткие прибыльные трейды	TesterStatistics
STAT_PROFIT_TRADES	Прибыльные трейды	TesterStatistics
STAT_PROFITTRADES_AVGCON	Средняя длина прибыльной серии трейдов	TesterStatistics
STAT_RECOVERY_FACTOR	Фактор восстановления - отношение STAT_PROFIT/STAT_BALANCE_DD	TesterStatistics
STAT_SHARPE_RATIO	Коэффициент Шарпа	TesterStatistics
STAT_SHORT_TRADES	Короткие трейды	TesterStatistics
STAT_TRADES	Количество трейдов	TesterStatistics
STAT_WITHDRAWAL	Количество выведенных со счета средств	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
STO_CLOSECLOSE	Построение по ценам Close/Close	Ценовые константы
STO_LOWHIGH	Построение по ценам Low/High	Ценовые константы
STYLE_DASH	Прерывистая линия	Стили рисования
STYLE_DASHDOT	Штрих-пунктирная линия	Стили рисования
STYLE_DASHDOTDOT	Штрих - две точки	Стили рисования
STYLE_DOT	Пунктирная линия	Стили рисования
STYLE_SOLID	Сплошная линия	Стили рисования
SUNDAY	Воскресенье	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SYMBOL_ASK	Ask - лучшее предложение на покупку	SymbolInfoDouble
SYMBOL_ASKHIGH	Максимальный Ask за день	SymbolInfoDouble
SYMBOL_ASKLOW	Минимальный	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	Ask за день	
SYMBOL_BANK	Источник текущей котировки	SymbolInfoString
SYMBOL_BASIS	Имя базового актива для производного инструмента	SymbolInfoString
SYMBOL_BID	Bid - лучшее предложение на продажу	SymbolInfoDouble
SYMBOL_BIDHIGH	Максимальный Bid за день	SymbolInfoDouble
SYMBOL_BIDLOW	Минимальный Bid за день	SymbolInfoDouble
SYMBOL_CALC_MODE_CFD	CFD mode - расчет залога и прибыли для CFD	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDINDEX	CFD index mode - расчет залога и прибыли для CFD	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	на индексы	
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD Leverage mode - расчет залога и прибыли для CFD при торговле с плечом	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES	Futures mode - расчет залога и прибыли для торговли фьючерсными контрактами на бирже	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS Futures mode - расчет залога и прибыли для торговли фьючерсными контрактами на FORTS.	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_STOCKS	Exchange mode - расчет залога и прибыли	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	для торговли ценными бумагами и на бирже	
SYMBOL_CALC_MODE_FOREX	Forex mode - расчет прибыли и маржи для Форекс	SymbolInfoInteger
SYMBOL_CALC_MODE_FUTURES	Futures mode - расчет залога и прибыли для фьючерсов	SymbolInfoInteger
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - инструмент используется в качестве неторгуемого актива на торговом счете.	SymbolInfoInteger
SYMBOL_CURRENCY_BASE	Базовая валюта инструмента	SymbolInfoString
SYMBOL_CURRENCY_MARGIN	Валюта в которой вычисля	SymbolInfoString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ется залоговые средства	
SYMBOL_CURRENCY_PROFIT	Валюта прибыли	SymbolInfoString
SYMBOL_DESCRIPTION	Строковое описание символа	SymbolInfoString
SYMBOL_DIGITS	Количество знаков после запятой	SymbolInfoInteger
SYMBOL_EXPIRATION_DAY	Ордер действителен до конца дня	SymbolInfoInteger
SYMBOL_EXPIRATION_GTC	Ордер действителен неограниченно по времени до явной его отмены	SymbolInfoInteger
SYMBOL_EXPIRATION_MODE	Флаги разрешенных режимов истечения ордера	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
SYMBOL_EXPIRATION_SPECIFIED	Срок истечения указывается в ордере	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED_DAY	День истечения указывается в ордере	SymbolInfoInteger
SYMBOL_EXPIRATION_TIME	Дата окончания торгов по инструменту (обычно используется для фьючерсов)	SymbolInfoInteger
SYMBOL_FILLING_FOK	Данная политика исполнения означает, что ордер может быть исполнен исключительно в указанном объеме. Если на рынке в данный момент	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	не присутствует достаточного объема финансового инструмента, то ордер не будет исполнен. Необходимый объем может быть составлен из нескольких предложений, доступных в данный момент на рынке.	
SYMBOL_FILLING_IOC	В данном случае трейдер соглашается совершить сделку по максимально доступному на рынке объему	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	в пределах указанного в ордере. В случае невозможности полного исполнения ордер будет исполнен на доступный объем, а неисполненный объем ордера будет отменен. Возможность использования IOC ордеров определяется на торговом сервере.	
SYMBOL_FILLING_MODE	Флаги разрешенных режимов заливки ордера	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
SYMBOL_ISIN	Имя торгового символа в системе международных идентификационных кодов ценных бумаг — ISIN (International Securities Identification Number). Международный идентификационный код ценной бумаги — это 12-разрядный буквенно-цифровой код, однозначно идентифицирующий ценную бумагу. Наличие данного свойств	SymbolInfoString

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	а символа определяется на стороне торгового сервера.	
SYMBOL_LAST	Цена, по которой совершена последняя сделка	SymbolInfoDouble
SYMBOL_LASTHIGH	Максимальный Last за день	SymbolInfoDouble
SYMBOL_LASTLOW	Минимальный Last за день	SymbolInfoDouble
SYMBOL_MARGIN_INITIAL	Начальная (инициирующая) маржа обозначает размер необходимых залоговых средств в маржинальной валюте для открытия позиции	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	объемом в один лот. Используется при проверке средств клиента при входе в рынок.	
SYMBOL_MARGIN_MAINTENANCE	Поддерживающая маржа по инструменту. В случае если задана - указывает размер маржи в маржинальной валюте инструмента, удерживаемой с одного лота. Используется при проверке средств клиента при изменении	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	состояния счета клиента. Если поддерживающая маржа равна 0, то используется начальная маржа.	
SYMBOL_OPTION_MODE	Тип опциона	SymbolInfoInteger
SYMBOL_OPTION_MODE_EUROPEAN	Европейский тип опциона - может быть погашен только в указанную дату (дата истечения срока, дата исполнения, дата погашения)	SymbolInfoInteger
SYMBOL_OPTION_MODE_AMERICAN	Американский тип опциона - может быть погашен в любой день до истечен	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ия срока опциона. Для такого типа задаётся период, в течение которого покупатель может исполнить данный опцион	
SYMBOL_OPTION_RIGHT	Право опциона (Call/Put)	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_CALL	Опцион, дающий право купить актив по фиксированной цене	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_PUT	Опцион, дающий право продать актив по фиксированной цене	SymbolInfoInteger
SYMBOL_OPTION_STRIKE	Цена исполнения опциона. Это цена, по	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	которой покупатель опциона может купить (при опционе Call) или продать (при опционе Put) базовый актив, а продавец опциона соответственно обязан продать или купить соответствующее количество базового актива.	
SYMBOL_ORDER_LIMIT	Разрешены лимитные ордера (Buy Limit и Sell Limit)	SymbolInfoInteger
SYMBOL_ORDER_MARKET	Разрешены рыночные ордера	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	(Buy и Sell)	
SYMBOL_ORDER_MODE	Флаги разрешенных типов ордера	SymbolInfoInteger
SYMBOL_ORDER_SL	Разрешена установка Stop Loss	SymbolInfoInteger
SYMBOL_ORDER_STOP	Разрешены стоп-ордера (Buy Stop и Sell Stop)	SymbolInfoInteger
SYMBOL_ORDER_STOP_LIMIT	Разрешены стоп-лимит ордера (Buy Stop Limit и Sell Stop Limit)	SymbolInfoInteger
SYMBOL_ORDER_TP	Разрешена установка Take Profit	SymbolInfoInteger
SYMBOL_PATH	Путь в дереве символов	SymbolInfoString
SYMBOL_POINT	Значение одного пункта	SymbolInfoDouble
SYMBOL_SELECT	Признак того,	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	что символ выбран в Market Watch	
SYMBOL_SESSION_AW	Средневзвешенная цена сессии	SymbolInfoDouble
SYMBOL_SESSION_BUY_ORDERS	Общее число ордеров на покупку в текущий момент	SymbolInfoInteger
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Общий объём ордеров на покупку в текущий момент	SymbolInfoDouble
SYMBOL_SESSION_CLOSE	Цена закрытия сессии	SymbolInfoDouble
SYMBOL_SESSION_DEALS	Количество сделок в текущей сессии	SymbolInfoInteger
SYMBOL_SESSION_INTEREST	Суммарный объём открытых позиций	SymbolInfoDouble
SYMBOL_SESSION_OPEN	Цена открытия сессии	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
SYMBOL_SESSION_PRICE_LIMIT_MAX	Максимально допустимое значение цены на сессию	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MIN	Минимально допустимое значение цены на сессию	SymbolInfoDouble
SYMBOL_SESSION_PRICE_SETTLEMENT	Цена поставки на текущую сессию	SymbolInfoDouble
SYMBOL_SESSION_SELL_ORDERS	Общее число ордеров на продажу в текущий момент	SymbolInfoInteger
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Общий объём ордеров на продажу в текущий момент	SymbolInfoDouble
SYMBOL_SESSION_TURNOVER	Суммарный оборот в текущую сессию	SymbolInfoDouble
SYMBOL_SESSION_VOLUME	Суммарный	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	объём сделок в текущую сессию	
SYMBOL_SPREAD	Размер спреда в пунктах	SymbolInfoInteger
SYMBOL_SPREAD_FLOAT	Признак плавающего спреда	SymbolInfoInteger
SYMBOL_START_TIME	Дата начала торгов по инструменту (обычно используется для фьючерсов)	SymbolInfoInteger
SYMBOL_SWAP_LONG	Значение свопа в покупку	SymbolInfoDouble
SYMBOL_SWAP_MODE	Модель расчета свопа	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Свопы начисляются в деньгах в валюте депозита клиента	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Свопы начисля	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ются в деньгах в маржинальной валюте символа	
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Свопы начисляются в деньгах в базовой валюте символа	SymbolInfoInteger
SYMBOL_SWAP_MODE_DISABLED	Нет свопов	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Свопы начисляются в годовых процентах от цены инструмента на момент расчета свопа(банковский режим - 360 дней в году)	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_OPEN	Свопы начисляются в годовых процентах от цены открытия позиции	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	по символу (банковский режим - 360 дней в году)	
SYMBOL_SWAP_MODE_POINTS	Свопы начисляются в пунктах	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_BID	Свопы начисляются переоткрытием позиции. В конце торгового дня позиция принудительно закрывается. На следующий день позиция переоткрывается по текущей цене Bid +/- указанное количество пунктов (в параметрах SYMBOL	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	_SWAP_LONG и SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Свопы начисляются переоткрытием позиции. В конце торгового дня позиция принудительно закрывается. На следующий день позиция переоткрывается по цене закрытия +/- указанное количество пунктов (в параметрах SYMBOL_SWAP_LONG и SYMBOL_SWAP_SHORT)	SymbolInfoInteger
SYMBOL_SWAP_ROLLOVER3DAYS	День недели	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	для начисления тройного свопа	
SYMBOL_SWAP_SHORT	Значение свопа в продажу	SymbolInfoDouble
SYMBOL_TICKS_BOOKDEPTH	Максимальное количество показываемых заявок в стакане. Для инструментов, не имеющих очереди заявок, значение равно 0	SymbolInfoInteger
SYMBOL_TIME	Время последней котировки	SymbolInfoInteger
SYMBOL_TRADE_CALC_MODE	Способ вычисления стоимости контракта	SymbolInfoInteger
SYMBOL_TRADE_CONTRACT_SIZE	Размер торгового контракта	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
SYMBOL_TRADE_EXECUTION_EXCHANGE	Биржевое исполнение	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_INSTANT	Торговля по потоковым ценам	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_MARKET	Исполнение ордеров по рынку	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_REQUEST	Торговля по запросу	SymbolInfoInteger
SYMBOL_TRADE_EXEMODE	Режим заключения сделок	SymbolInfoInteger
SYMBOL_TRADE_FREEZE_LEVEL	Дистанция заморозки торговых операций (в пунктах)	SymbolInfoInteger
SYMBOL_TRADE_MODE	Тип исполнения ордеров	SymbolInfoInteger
SYMBOL_TRADE_MODE_CLOSEONLY	Разрешены только операции закрытия позиций	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
SYMBOL_TRADE_MODE_DISABLED	Торговля по символу запрещена	SymbolInfoInteger
SYMBOL_TRADE_MODE_FULL	Нет ограничений на торговые операции	SymbolInfoInteger
SYMBOL_TRADE_MODE_LONGONLY	Разрешены только покупки	SymbolInfoInteger
SYMBOL_TRADE_MODE_SHORTONLY	Разрешены только продажи	SymbolInfoInteger
SYMBOL_TRADE_STOPS_LEVEL	Минимальный отступ в пунктах от текущей цены закрытия для установки Stop ордеров	SymbolInfoInteger
SYMBOL_TRADE_TICK_SIZE	Минимальное изменение цены	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE	Значение SYMBOL_TRADE_TICK_VALUE_PROFIT	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
SYMBOL_TRADE_TICK_VALUE_LOSS	Рассчитанная стоимость тика для убыточной позиции	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_PROFIT	Рассчитанная стоимость тика для прибыльной позиции	SymbolInfoDouble
SYMBOL_VOLUME	Volume - объем в последней сделке	SymbolInfoInteger
SYMBOL_VOLUME_LIMIT	Максимально допустимый для данного символа совокупный объем открытой позиции и отложенных ордеров в одном направлении (покупка или продажа). Наприм	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ер, при ограничении в 5 лотов можно иметь открытую позицию на покупку объемом 5 лотов и выставить отложенный ордер Sell Limit объемом 5 лотов. Но при этом нельзя выставить отложенный ордер Buy Limit (поскольку совокупный объем в одном направлении превысит ограничение) или выстави	

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ть Sell Limit объемом более 5 лотов.	
SYMBOL_VOLUME_MAX	Максимальный объем для заключения сделки	SymbolInfoDouble
SYMBOL_VOLUME_MIN	Минимальный объем для заключения сделки	SymbolInfoDouble
SYMBOL_VOLUME_STEP	Минимальный шаг изменения объема для заключения сделки	SymbolInfoDouble
SYMBOL_VOLUMEHIGH	Максимальный Volume за день	SymbolInfoInteger
SYMBOL_VOLUMELOW	Минимальный Volume за день	SymbolInfoInteger
TENKANSEN_LINE	Линия Tenkan-sen	Линии индикаторов
TERMINAL_BUILD	Номер билда запущен	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ного терминала	
TERMINAL_CODEPAGE	Номер кодовой страницы языка, установленного в клиентском терминале	TerminalInfoInteger
TERMINAL_COMMONDATA_PATH	Общая папка всех клиентских терминалов, установленных на компьютере	TerminalInfoString
TERMINAL_COMMUNITY_ACCOUNT	Флаг наличия авторизационных данных MQL5.community в терминале	TerminalInfoInteger
TERMINAL_COMMUNITY_BALANCE	Баланс пользователя в MQL5.community	TerminalInfoDouble

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
TERMINAL_COMMUNITY_CONNECTION	Наличие подключения к MQL5.community	TerminalInfoInteger
TERMINAL_COMPANY	Имя компании	TerminalInfoString
TERMINAL_CONNECTED	Наличие подключения к торговому серверу	TerminalInfoInteger
TERMINAL_CPU_CORES	Количество процессоров в системе	TerminalInfoInteger
TERMINAL_DATA_PATH	Папка, в которой хранятся данные терминала	TerminalInfoString
TERMINAL_DISK_SPACE	Объем свободной памяти на диске для папки MQL5\Files терминала (агента), в MB	TerminalInfoInteger
TERMINAL_DLLS_ALLOWED	Разрешение на использ	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ование DLL	
TERMINAL_EMAIL_ENABLED	Разрешение на отправку писем с использованием SMTP-сервера и логина, указанных в настройках терминала	TerminalInfoInteger
TERMINAL_FTP_ENABLED	Разрешение на отправку отчетов по FTP на указанный сервер для указанного в настройках терминала торгового счета	TerminalInfoInteger
TERMINAL_LANGUAGE	Язык терминала	TerminalInfoString
TERMINAL_MAXBARS	Максимальное количес	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	тво баров на графике	
TERMINAL_MEMORY_AVAILABLE	Размер свободной памяти процесса терминала (агента) в MB	TerminalInfoInteger
TERMINAL_MEMORY_PHYSICAL	Размер физической памяти в системе, в MB	TerminalInfoInteger
TERMINAL_MEMORY_TOTAL	Размер памяти, доступной процессу терминала (агента), в MB	TerminalInfoInteger
TERMINAL_MEMORY_USED	Размер памяти, использованной терминалом (агентом), в MB	TerminalInfoInteger
TERMINAL_MQID	Флаг наличия MetaQuotes ID для отправк	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	и Push-уведомлений	
TERMINAL_NAME	Имя терминала	TerminalInfoString
TERMINAL_NOTIFICATIONS_ENABLED	Разрешение на отправку уведомлений на смартфон	TerminalInfoInteger
TERMINAL_OPENCL_SUPPORT	Версия поддерживаемой OpenCL в виде 0x00010002 = 1.2. "0" означает, что OpenCL не поддерживается	TerminalInfoInteger
TERMINAL_PATH	Папка, из которой запущен терминал	TerminalInfoString
TERMINAL_PING_LAST	Последнее известное значение пинга до торгового	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	сервера в микросекундах. В одной секунде миллион микросекунд	
TERMINAL_SCREEN_DPI	Разрешающая способность вывода информации на экран	TerminalInfoInteger
TERMINAL_TRADE_ALLOWED	Разрешение на торговлю	TerminalInfoInteger
TERMINAL_X64	Признак "64 битный терминал"	TerminalInfoInteger
THURSDAY	Четверг	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TRADE_ACTION_DEAL	Установить торговый ордер на немедленное совершение сделки с указанными параметрами (поставить	MqlTradeRequest

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	рыночный ордер)	
TRADE_ACTION_MODIFY	Изменить параметры ранее установленного торгового ордера	MqlTradeRequest
TRADE_ACTION_PENDING	Установить торговый ордер на совершение сделки при указанных условиях (отложенный ордер)	MqlTradeRequest
TRADE_ACTION_REMOVE	Удалить ранее выставленный отложенный торговый ордер	MqlTradeRequest
TRADE_ACTION_SLTP	Изменить значения Stop Loss и Take Profit у открыто	MqlTradeRequest

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	й позиции	
TRADE_RETCODE_CANCEL	Запрос отменен трейдером	MqlTradeResult
TRADE_RETCODE_CLIENT_DISABLES_AT	Автотрейдинг запрещен клиентским терминалом	MqlTradeResult
TRADE_RETCODE_CONNECTION	Нет соединения с торговым сервером	MqlTradeResult
TRADE_RETCODE_DONE	Заявка выполнена	MqlTradeResult
TRADE_RETCODE_DONE_PARTIAL	Заявка выполнена частично	MqlTradeResult
TRADE_RETCODE_ERROR	Ошибка обработки запроса	MqlTradeResult
TRADE_RETCODE_FROZEN	Ордер или позиция заморожены	MqlTradeResult
TRADE_RETCODE_INVALID	Неправильный запрос	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
TRADE_RETCODE_INVALID_EXPIRATION	Неверная дата истечения ордера в запросе	MqlTradeResult
TRADE_RETCODE_INVALID_FILL	Указан неподдерживаемый тип исполнения ордера по остатку	MqlTradeResult
TRADE_RETCODE_INVALID_ORDER	Неверный или запрещённый тип ордера	MqlTradeResult
TRADE_RETCODE_INVALID_PRICE	Неправильная цена в запросе	MqlTradeResult
TRADE_RETCODE_INVALID_STOPS	Неправильные стопы в запросе	MqlTradeResult
TRADE_RETCODE_INVALID_VOLUME	Неправильный объем в запросе	MqlTradeResult
TRADE_RETCODE_LIMIT_ORDERS	Достигнут лимит на количество отложенных ордеров	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
TRADE_RETCODE_LIMIT_VOLUME	Достигнут лимит на объем ордеров и позиций для данного символа	MqlTradeResult
TRADE_RETCODE_LOCKED	Запрос заблокирован для обработки	MqlTradeResult
TRADE_RETCODE_MARKET_CLOSED	Рынок закрыт	MqlTradeResult
TRADE_RETCODE_NO_CHANGES	В запросе нет изменений	MqlTradeResult
TRADE_RETCODE_NO_MONEY	Нет достаточных денежных средств для выполнения запроса	MqlTradeResult
TRADE_RETCODE_ONLY_REAL	Операция разрешена только для реальных счетов	MqlTradeResult
TRADE_RETCODE_ORDER_CHANGED	Состояние ордера	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	изменилось	
TRADE_RETCODE_PLACED	Ордер размещен	MqlTradeResult
TRADE_RETCODE_POSITION_CLOSED	Позиция с указанным POSITION_IDENTIFIER уже закрыта	MqlTradeResult
TRADE_RETCODE_PRICE_CHANGED	Цены изменились	MqlTradeResult
TRADE_RETCODE_PRICE_OFF	Отсутствуют котировки для обработки запроса	MqlTradeResult
TRADE_RETCODE_REJECT	Запрос отвергнут	MqlTradeResult
TRADE_RETCODE_REQUOTE	Реквота	MqlTradeResult
TRADE_RETCODE_SERVER_DISABLES_AT	Автотрейдинг запрещен сервером	MqlTradeResult
TRADE_RETCODE_TIMEOUT	Запрос отменен по истечению времени	MqlTradeResult
TRADE_RETCODE_TOO_MANY_REQUESTS	Слишком	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	частые запросы	
TRADE_RETCODE_TRADE_DISABLED	Торговля запрещена	MqlTradeResult
TRADE_TRANSACTION_DEAL_ADD	Добавление сделки в историю. Осуществляется в результате исполнения ордера или проведения операций с балансом счета.	MqlTradeTransaction
TRADE_TRANSACTION_DEAL_DELETE	Удаление сделки из истории. Возможны ситуации, когда ранее исполненная сделка удаляется на сервере. Например,	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	сделка была удалена во внешней торговой системе (бирже), куда она была выведена брокером.	
TRADE_TRANSACTION_DEAL_UPDATE	Изменение сделки в истории. Возможны ситуации, когда ранее исполненная сделка изменяется на сервере. Например, сделка была изменена во внешней торговой системе (бирже), куда она была	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	выведена брокером.	
TRADE_TRANSACTION_HISTORY_ADD	Добавление ордера в историю в результате исполнения или отмены.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_DELETE	Удаление ордера из истории ордеров. Данный тип предусмотрен для расширения функциональности на стороне торгового сервера.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_UPDATE	Изменение ордера, находящегося в истории ордеров.	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	Данный тип предусмотрен для расширения функциональности на стороне торгового сервера.	
TRADE_TRANSACTION_ORDER_ADD	Добавление нового открытого ордера.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_DELETE	Удаление ордера из списка открытых. Ордер может быть удален из открытых в результате выставления соответствующего запроса либо в результате	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	исполнения (заливки) и переноса в историю.	
TRADE_TRANSACTION_ORDER_UPDATE	Изменение открытого ордера. К данным изменениям относятся не только явные изменения со стороны клиентского терминала или торгового сервера, но также и изменение его состояния при выставлении (например, переход из состояния ORDER_STATE_S	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	TARTED в ORDER_STATE_PLACED или из ORDER_STATE_PLACED в ORDER_STATE_PARTIAL и т.д.).	
TRADE_TRANSACTION_POSITION	Изменение позиции, не связанное с исполнением сделки. Данный тип транзакции свидетельствует именно о том, что позиция была изменена на стороне торгового сервера. У позиции может быть изменен объем, цена открыти	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	я, а также уровни Stop Loss и Take Profit. Информация об изменениях передается в структуре MqlTradeTransaction через обработчик OnTradeTransaction. Изменение позиции (добавление, изменение или ликвидация) в результате совершения сделки не влечет за собой появление транзакции TRADE_TRANSACTION_P	

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	OSITION.	
TRADE_TRANSACTION_REQUEST	Уведомление о том, что торговый запрос обработан сервером, и результат его обработки получен. Для транзакций данного типа в структуре MqlTradeTransaction необходимо анализировать только одно поле - type (тип транзакции). Для получения дополнительной информации необходимо анализи	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	ровать второй и третий параметры функции OnTradeTransaction (request и result).	
TUESDAY	Вторник	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	Максимальное значение, которое может быть предста	Константы числовых типов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
	влено типом uchar	
UINT_MAX	Максимальное значение, которое может быть представлено типом uint	Константы числовых типов
ULONG_MAX	Максимальное значение, которое может быть представлено типом ulong	Константы числовых типов
UPPER_BAND	Верхняя граница	Линии индикаторов
UPPER_HISTOGRAM	Верхняя гистограмма	Линии индикаторов
UPPER_LINE	Верхняя линия	Линии индикаторов
USHORT_MAX	Максимальное значение, которое может быть представлено типом ushort	Константы числовых типов

© 2000-2025, MetaQuotes Ltd.

Константа	Описание	Использование
VOLUME_REAL	Торговый объем	Ценовые константы
VOLUME_TICK	Тиковый объем	Ценовые константы
WEDNESDAY	Среда	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
WHOLE_ARRAY	Означает количество элементов, оставшееся до конца массива, то есть, будет обработан весь массив	Прочие константы
WRONG_VALUE	Константа может неявно приводиться к типу любого перечисления.	Прочие константы

© 2000-2025, MetaQuotes Ltd.